AFRL-IF-RS-TR-2007-20
Final Technical Report
January 2007

CONTROL RECONFIGURATION OF COMMAND AND CONTROL SYSTEMS

The Research Foundation of State University of New York

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Rome Research Site Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-IF-RS-TR-2007-20 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
TIMOTHY E. BUSCH Work Unit Manager	JAMES W. CUSACK, Chief Information Systems Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JAN 2007	Final	Sep 02 – Sep 06

4. TITLE AND SUBTITLE

CONTROL RECONFIGURATION OF COMMAND AND CONTROL SYSTEMS

5a. CONTRACT NUMBER
F30602-02-C-0225

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61102F/62702F

6. AUTHOR(S)

N. Eva Wu

5d. PROJECT NUMBER
2304

5e. TASK NUMBER
CR

5f. WORK UNIT NUMBER
CS

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Research Foundation of State University of New York
35 State St.
Albany NY 12207

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFSB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2007-20

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 07-026

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Currently, an air operation center (AOC) for a major regional conflict is composed of more than 400 personnel, hundreds of computer servers and an extensive communication infrastructure. So, in addition to the goals of achieving a faster, more real time response, there is also a desire to reduce the manning and equipment associated with the endeavor. This means that the management of redundancy must be optimized. An important consideration in the design and fielding of such systems is its capacity to accommodate faults through control reconfiguration using either direct or analytic redundancy. The latter relies on exploiting the inherent dynamic and static relationship of the system variables, and having the advantage of most efficient use of the components. This research applied the concepts of control reconfigurability to C2 systems modeled as stochastic discrete event systems.

15. SUBJECT TERMS
Command and Control, Robust Control, Markov Decision Processes, Reconfigurability, Information Systems

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Timothy E. Busch
U	U	U	UL	63	19b. TELEPHONE NUMBER (*Include area code*)

Executive Summary

Currently, an air operation center (AOC) for a major regional conflict is composed of more than 400 personnel, hundreds of computer servers and an extensive communication infrastructure. So in addition to the goals of achieving a faster, more real time response, there is also a desire to reduce the manning and equipment associated with the endeavor. This means that the management of redundancy must be optimized. An important consideration in the design and fielding of such systems is its capacity to accommodate faults through control reconfiguration using either direct or analytic redundancy. The latter relies on exploiting the inherent dynamic and static relationship of the system variables, and having the advantage of most efficient use of the components. The proposed research seeks to apply the concepts of control reconfigurability to C2 systems modeled as stochastic discrete event systems.

This report contains ten self-contained sections drawn from eight fully refereed conference papers and two reports produced over the past four years. The sections are organized into three chapters: Reconfigurability Concepts and C2-AO Relationship, Feedback and Reliability of Wireless Sensor Networks in C2, and Supervisory Control and Performance Analysis of C2 Database. This effort has produced three MSEE Theses (Metzler, Kussard, and Ruschmann), one additional conference paper under review, one accepted journal paper, and one invited journal paper to be submitted within this month.

Table of Contents

List of Figures

List of Tables

1. Strategic Reconfigurability in Air Operations

1.1. Problem description

In military air operations, the delivery speed and accuracy of modern weapons make swift decisions desirable, while advanced sensing and processing capabilities make them possible. These traits have been well recognized, and have directed the focus of some research activities onto closing the feedback control loops in air operations, despite many differing philosophies in modeling [12], [21], [11] and techniques for analysis and control [10], [17], [41]. Due to large uncertainties and computational burdens, however, it is difficult for feedback designs to be proactive [11], which requires accurate prediction far enough into the future. Though one can establish attrition models in the form of Markov chains from the first principles [21], such models provide reliable predictions only for a general population of air operations. They capture little dynamic effect of an individual tactical operation on the execution of a strategic plan.

The first section of this report is intended to address the issue of resilience in air operations. The notion of strategic reconfigurability is introduced that measures the ability of a military air operation to successfully respond to contingencies. The intended goal requires a new modeling framework that is geared toward the closed-loop control of the air operation, and allows a closer scrutiny on how a tactical execution affects the evolution of a strategic plan. This section proposes to model an air operation as a two-level, two-timescale feedback system. In this framework, coverage of state transitions in stochastic discrete state model with relatively a low resolution serves to effect the supervisory control of the strategic operation, while the remaining detail in the air operation is represented by a continuous state model with bounded disturbances. The former will be called a strategic model, and the latter will be called a tactical model. The overall model can be regarded as a hybrid dynamic system. Unlike most existing efforts in hybrid system modeling [2], however, our interest is in capturing the interactions between the strategic and the tactical models. In particular, strategic states enter the tactical model as symbolic parameters, while the dynamic effects of the tactical operation on the strategic model are represented by a set of state transition coverage parameters. The two-level framework makes it possible to describe a complex air operation with a model of much reduced complexity, provided that the two sets of parameters project well the collective effects of the two models on each other.

Among many possible benefits in using the two-level air operation model, this section focuses on making use of the separability of the two-level model in the investigation of strategic reconfigurability. The notion of strategic reconfigurability is particular with respect to transitions of strategic states that are controllable by Blue. The transition coverage allows our study on reconfigurability to include the dynamics associated with information acquisition and processing, and the risks associated with control policy making. Our investigation can be confined at the strategic level with the effects of tactical operation observable through a set of state transition coverage parameters.

The section is organized as follows. Subsection B describes the two-level framework and the role of transition coverage in it. Subection C defines the notions of specific and strategic reconfigurability, and explains their utility. A summary is presnted drawn in Subection D.

1.2. Coverage of transitions in a strategic model

A strategic model refers to the mathematical description of the evolution of a strategic plan in an air operation. It takes the form of a discrete state and continuous (discrete) time Markov process (chain) [20]. The fundamental assumption of a Markov process is that the probability that a system will undergo a transition from one state to another state depends only on the current state of the system and not any previous states the system may have experienced. To specify a Markov model, it is necessary to identify (i) a state space $\{\mathcal{X}\}$, (ii) a set of initial state probabilities $\{p_x(0), x \in \mathcal{X}\}$, and (iii) a set of state transition rates $\{\lambda_{x,x'}(t), x, x' \in \mathcal{X}\}$ from the current state x to the next state x' [8]. A Markov process is said to be homogeneous if all state transition rates are independent of time. Strategic planning for an air operation can be regarded as the process of determining both the state space and the state transition rates of a Markov process under some constraints which include limited resources, laws of physics, etc.

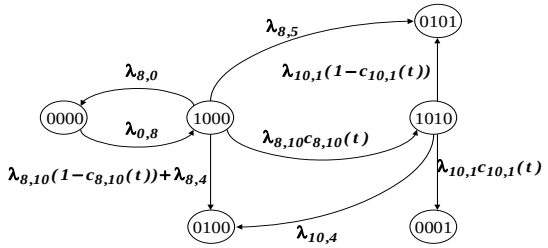


Figure 1.1 A low resolution strategic model

A transition coverage associated with a transition from x to x' is the conditional probability that the intended transition in fact occurs given that a triggering event has arrived. It is denoted by $c^u_{x,x'}(t)$, where u indicates its dependence on the control policy used in the tactical operation. It provides a means to separate

the handling of an event from the occurrence of the event. Figure 1.1 shows a sample strategic model at a very low resolution in the form of a rate diagram of a Markov process. The simple model is chosen to highlight our approach. In practice, the resolution is determined by our ability and willingness to deal with complexity, and by the resolution of the information available to us. It can be seen that a transition coverage serves to effectively modify the corresponding transition rate via $\lambda_{x,x'} c^u_{x,x'}(t)$. This modification is justified by a decomposition property of a Poisson process [22], where the arrival of a triggering event may lead to two (multiple) transitions with conditional probabilities which we call transition coverage and complementary coverage, respectively.

The state space of the strategic model in Figure 1.1 is determined by the composition of four binary states: (Blue threatened, Blue defeated, Red targeted, Red defeated). A state of (True, False, True, False) can be represented by $x = 1010$ in binary. The meanings of the remaining states can be similarly explained. The possible states that do not show up in the diagram are ones that have identically zero state probabilities. The links that are missing between a pair of states correspond to transitions that have zero transition rates. States 0000, 1000, and 1010 in Figure 1.1 are transient states and states 0100, 0101, and 0001 are absorbing states. Depending on whether preserving the Blue assets besides destroying the Red assets is also part of the mission of an air operation, the set of desirable outcomes for Blue can be one of $\{0101, 0001\}$ and $\{0001\}$.

The discussion on modeling in this section is focused on transition coverage rather than transition rates. Therefore it is assumed that state transition rates have been determined for the model. The reader having an interest in how transition rates in air operations are determined is referred to the two sample models [21], [41] in Table S1 which also intends to show the differences of the two models. It is easy to determine which transitions are controllable by Blue. The controllable transitions are those having a transition coverage attached to them. When a strategic plan is fully carried out, all values of transition coverage associated with controllable transitions are set to 1 identically. When a planned action associated with a transition is totally ignored, the value of the corresponding transition coverage is set to 0. In this regard, transition coverage serves as a supervisory control agent at the strategic level, which is dictated by the extent of success in carrying out the tactical execution.

Reference	Jelinek [21]	Wohletz et al. [41]
Discrete states	Quantitative numbers of live units making up the Blue and Red assets, each with a certain amount of remaining ammunition	Qualitative symbols formed from parallel composition of smaller sets of states, such as that from a Blue aircraft (base, ingress, egress, dead), a Red target (known, unknown, dead), etc.
Transition rates	Derived from the first principles in terms of the number of surviving units, number of targets in an area, time to search for a target, time to aim weapon at it, weapon lethality, etc.	Derived from Poisson clock structures that define the arrivals of triggering events, such as (launch, threat engage, target engage, land) for aircraft, etc., with specified interarrival mean times
Model utility	Outcome prediction of air operations	Decisions in and control of air operations

Table S1 Two sample Markov models

We raise the following questions. What constitutes an appropriate measure for the effect of a transition coverage on the overall air operation? What factors determine the effectiveness of transition coverage in a controllable transition? The answer to the first question will be attempted in the next subsection, and a qualitative answer to the second question is given in the current subsection.

One fundamental assumption in strategic planning for air operations that involve feedback is the knowledge of the current strategic state. This information is often deficient in a real air operation. Therefore, risks exist in executing a control policy that attempts to drive the air operation into a desirable next strategic state. These risks can now be properly represented using transition coverage in the strategic model so that their effects can be studied. Such a study is expected to help modifiy the strategic plan and determine control policies that minimize certain risks.

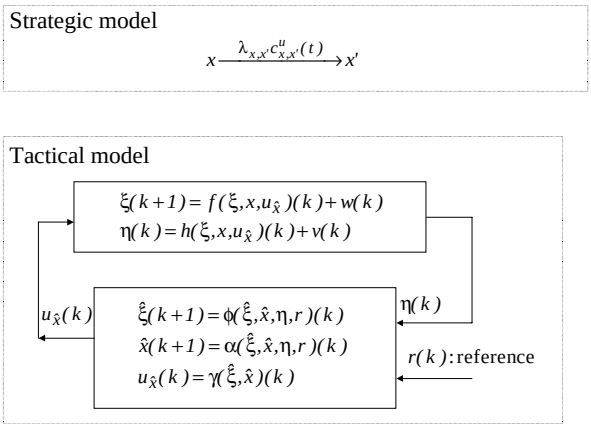


Figure 1.2 A two-level model of an air operation

Our answer to the second question begins with the description of a tactical operation model. A tactical model refers to a closed-loop control system which governs

the execution of an air operation. A tactical model is a continuous-state dynamic process with bounded disturbances. Figure 1.2 shows how it is related to a (overly simplfied) strategic model represented by a single transition. Note that the tactical states do not include those of the opposing force (Red). With the use of a smaller timescale than that of the strategic model, a tactical model is able to describe the behavior of Blue in greater detail. The tactical state vector may contain in its components, for example, the strengths of Blue assets, the rates of change of the strengths, their geographic locations, rates of change of locations, etc. In this fashion, a less likely strategic state that enters the tactical model as a deterministic symbolic or numerical parameter will be regarded as equally important in the tactical operation once it has arrived. The air operation at the tactical level is responsible for generating two sets of signals. One is an estimate of the strategic state $\hat{x}$, and another is a corresponding control $u_{\hat{x}}$ to drive the tactical state ξ to wherever desirable. Suppose with respect to each strategic state a prescribed control performance threshold $\bar{\mathcal{T}}_x$ is set such that attained control performance J_x^u is required to exceed $\bar{\mathcal{T}}_x$. The transition coverage from x to x' can now be formalized as

$$c^u_{x,x'}(t) = \sum_{\hat{x}' \in \Omega_u} p(\hat{x}'|x')(t), \quad \Omega_u \equiv \{x \in \mathcal{X} | J_x^u \geq \bar{\mathcal{T}}_x\} \tag{1}$$

where $p(\hat{x}'|x')$ is the probability of estimate $\hat{x}'$ parameterized with respect to x'. For simplicity the estimate is assumed to be unbiased and consistent [7].

The models in the two levels together form a system which is hybrid not only in terms of discrete state versus continuous state descriptions, but also in terms of slower versus faster dynamics.

(9) indicates that $c^u_{x,x'}$ is functionally related to the required tactical level performance $\bar{\mathcal{T}}_{x'}$, the achieved tactical performance $J^u_{\hat{x}'}$ via using u, and the uncertainty description associated with estimate $\hat{x}'$ of strategic state x'. Transition coverage therefore summarizes the dynamic response of a tactical operation.

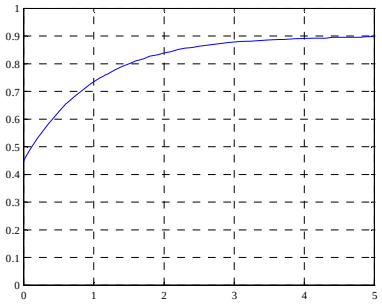


Figure 1.3 A typical temporal profile of a transition coverage

One aspect of transition coverage that has not been explicitly discussed is its dependence on time. The time dependence of coverage is mainly attributed to the time dependence of the conditional distribution of the estimate of a strategic state, as shown in (9). We make a loose argument that as information is acquired and processed, the distribution of $\hat{x}'|x'$ becomes more specific. Assume, for simplicity, that $\bar{\mathcal{T}}_{x'} = \bar{\mathcal{T}}$, $\forall x'$, and that the mean of the estimate of x' is in Ω_u, i.e.,

$$E_{\hat{x}'}\{\hat{x}'|x'\}(t) = \sum_{\hat{x}'} \hat{x} p(\hat{x}'|x')(t) = x' \in \Omega_u.$$

If increasing t eventually reduces the spread of $p(\hat{x}'|x')(t)$ to the extent that $p(\hat{x}'|x')(\infty) = \delta(\hat{x}' - x')$, then $c^u_{x,x'}(t) \overset{t\to\infty}{\to} 1$. This argument is intended to signify the general trend of $c^u_{x,x'}(t)$ as t increases. In particular, transition coverage is expressible by a three parameter representation $A(1 - \Delta e^{-t/\tau})$, as shown in Figure 1.3, where $0 < A \leq 1$ defines the final value of a transition coverage, $0 \leq \Delta \leq 1$ defines the total increment of the transition coverage over time, and $\tau > 0$ defines the rate of increase of the transition coverage. This expression is used to capture both the steady state and the transient tactical performance. In fact $c^u_{x,x'}(t)$ can be obtained in real-time by using (9) where, for example, the first and second moments of the probability distribution $p(\hat{x}'|x')(t)$ is estimated from the samples collected.

The simple example in Figure 1.1 is used to observe some rather dramatic effects of the level and the dynamics of coverage on winning probabilities at 100 hours into the air operation. The following data are used in producing the result in Table S2. $\lambda_{0,8} = 0.2$, $\lambda_{8,0} = 0.02$, $\lambda_{8,4} = 0.04$, $\lambda_{8,5} = 0.001$, $\lambda_{8,10} = 0.4$, $\lambda_{10,4} = 0.005$, $\lambda_{10,1} = 0.05$.

$c_{8,10}$, $c_{10,1}$	Blue wins	Red wins	Both lose
1, 1	0.82	0.18	0.00
0.5, 0.5	0.21	0.58	0.21
$.95(1 - .5e^{-t/5})$, $.95(1 - .5e^{-t/10})$	0.56	0.35	0.09

Table S2 Winning probabilities at $t = 100$ hours when transient state probabilities have died out.

1.3. Strategic reconfigurability

The analysis of the previous subsection has shown that transition coverage $c^u_{x,x'}(t)$ can serve to effect supervisory control of an air operation. In particular, a Markov decision problem [8] can be formulated for which an optimal supervisory control policy can be sought from an admissible set $\{c^u_{x,x'}(t)\}$. A trivial example would be the greedy policy $\max_{u\in\mathcal{U}} c^u_{x,\hat{x}'}(t)$ with cost functional $1 - c^u_{x,\hat{x}'}(t)$, where $\mathcal{U}$ is the set of admissible tactical control laws. $\mathcal{U}$ can be finite, countably infinite, or uncountable.

The purpose here, however, is not to solve a Markov decision problem, but to define a measure of the ability

of an air operation to respond favorably to contingencies. From the view of a discrete event dynamic system, a contingent condition means the arrival of one of multiple probable triggering events that generally has an unfavorable effect on Blue. Referring to Figure 1.1, it corresponds to a situation where multiple transitions out of a current state are probable. Therefore, both strategic planning and tactical execution must be designed to be capable of dealing with contingencies. To quantify such a capability a measure called a strategic reconfigurability is introduced.

The notion of control reconfigurability has been used for fault-tolerant control systems [52] for the purpose of measuring the ability of a process to allow performance restoration via control reconfiguration in the presence of faults. Control reconfigurability is the worst case minimal system Hankel singular value for a specific set of faulty conditions. Therefore, it involves both controllability and observability of the controlled process. The term strategic reconfigurability is adopted based on the rationale that the measure has indeed a close relationship to both the controllability and the observability of an air operation. First, only controllable transitions will be involved in the strategic reconfigurability measure. In addition, it is only through the transition coverage, control actions can be exerted onto the strategic operation, while a low transition coverage is largely attributed to lack of observability of the involved strategic state. Despite the similarity in terminology, strategic reconfigurability defined in this section is an entirely different quantity from control reconfigurability.

For a given strategic model, the set of state transition probabilities can be solved from the forward Kolmogorov equation

$$\dot{P}(t) = P(t)Q(t), P(0) = 1. \tag{2}$$

Therefore, transition rate matrix $Q(t)$ completely determines the set of transition probabilities. Let us revisit the strategic model of Figure 1.1 to obtain an explicit $Q(t)$. Ordering the states from absorbing to transient as $(0001, 0100, 0101, 0000, 1000, 1010)$, or $(1, 4, 5, 0, 8, 10)$ in decimal, for the strategic model, the 6×6 transition rate matrix $Q(t)$ can be found to be

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \lambda_{8,10}(1-c_{8,10}(t))+\lambda_{8,4} & \lambda_{8,5} \\ \lambda_{10,1}c_{10,1}(t) & \lambda_{10,4} & \lambda_{10,1}(1-c_{10,1}(t)) \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\lambda_{0,8} & \lambda_{0,8} & 0 \\ \lambda_{8,0} & -\lambda_{8,4}-\lambda_{8,0}-\lambda_{8,5}-\lambda_{8,10} & \lambda_{8,10}c_{8,10}(t) \\ 0 & 0 & -\lambda_{10,1}-\lambda_{10,4} \end{bmatrix}.$$

A canonical form for a strategic model will be used in the formal definition of strategic reconfigurability. The steps below can be followed to modify a given strategic state space to a canonical form. It can be checked that our sample strategic model has been put into the canonical form.

1) For a given strategic model, decompose the state space into the set of transient states and the set of absorbing states, i.e., $\mathcal{X} = \mathcal{X}_t \cup \mathcal{X}_a$.
2) For the set of transient states, identify the subset of controllable states from each of which at least one controllable transition stems, and denote the subset by $\mathcal{X}_t^c$.
3) Further decompose the set of absorbing states into the subset of desirable absorbing states and the subset of undesirable absorbing states, i.e., $\mathcal{X}_a = \mathcal{X}_a^d \cup \mathcal{X}_a^u$. The set of desirable absorbing states, and the set of undesirable absorbing states are dictated by the mission of the air operation.
4) Finally, reorganize the state space by aggregating all desirable absorbing states into one winning state x_w for Blue. Without loss of generality, the aggregated winning state is placed in the first entry of the strategic state vector.

Definition 1. Specific reconfigurability for $x \in \mathcal{X}_t^c$ is given by

$$\rho_x \equiv \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots \end{bmatrix} P(\infty) \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \\ 0 \end{bmatrix}, \tag{3}$$

where the only entry in the row vector on the left containing a 1 corresponds to where transient state x resides in the strategic state vector.

There is a specific reconfigurability for every $x \in \mathcal{X}_t^c$. Specific reconfigurabilities reflect an aspect of the structural property of a strategic model. In particular, the specific reconfigurability associated with state x quantifies the likelihood of winning the air operation upon entering the state. It also provides important information for real time tactical decisions and strategic re-planning. The structure of the strategic model and the transition coverage out of x are the two determining factors for the value of ρ_x.

Definition 2. Strategic reconfigurability is the arithmetic average of all specific reconfigurabilities associated with a given strategic model, i.e.,

$$\rho \equiv \frac{1}{|\mathcal{X}_t^c|} \sum_{x \in \mathcal{X}_t^c} \rho_x. \tag{4}$$

Strategic reconfigurability measures the ability to respond to contingencies of the overall air operation. Therefore, it can be used as a criterion to compare different strategic plans.

The two specific reconfigurabilities for state 1000 and state 1010 in the strategic model described in Figure

1.1 are

$$\begin{bmatrix} \rho_8 \\ \rho_{10} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} P(\infty) \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The strategic reconfigurability of the same model is

$$\rho = \frac{1}{2}\rho_8 + \frac{1}{2}\rho_{10}.$$

We now examine the effects of the two factors, i.e., strategic model structure and transition coverage, on the strategic reconfigurability.

Effect of strategic model structure. A modification of the strategic plan associated with Figure 1.1 is made. In the new plan Blue is always alert and ready. To keep the simplicity of the original model and make a meaningful comparison between the two strategic operations, only Red targeted in the original one of four binary states has been changed to Blue ready or offensive. This requires a careful strategic planning that fully exposes the strength of Blue. A changed strategic plan is shown in Figure 1.4. The reader can follow the description of Figure 1.1 to understand what the remaining changes should be.

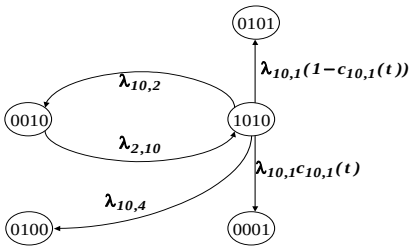


Figure 1.4 An alternative strategic model

Order the states from absorbing to transient as $(0001, 0100, 0101, 0010, 1010)$, or $(1, 4, 5, 2, 10)$ in decimal, for the model in Figure 1.4 to obtain a canonical representation of the model. Taking $\lambda_{2,10} = \lambda_{0,8} = 0.2$, $\lambda_{10,2} = \lambda_{8,0} = 0.02$, and the rest of parameters equal to the values given in the previous subsection, we obtain $\rho = 0.56$ for the original strategic plan, and $\rho = 0.71$ for the alternative strategic plan. The winning probabilities for Blue of the two plans are 0.57 (Blue defensive) and 0.76 (Blue offensive), respectively.

Effect of transition coverage. The previous subsection has described that the transition coverage represents an attempt to separate the handling of an event from the occurrence of the event. It is a means of capturing the dynamic process of tactical execution including both estimation and control, and that it can be manipulated to facilitate certain outcomes in an air operation. Since a specific reconfigurability is essentially the probability of winning starting at a specific controllable state, it is more informative of the effect of the transition coverage in the planning and the execution of an air operation.

For the strategic model in Figure 1.1, specific reconfigurabilities are calculated for the 2×3 parameters in transition coverage $c_{8,10}(t)$ and $c_{10,1}(t)$. The results are shown in Figure 1.5. The same state transition rates and nominal coverage parameters as those in Table S2 are used. Only a single parameter that represents the horizontal axis in each plot is varied, and the remaining 5 parameters are kept at the nominal values. The display of specific reconfigurability is accompanied by the corresponding winning probability of Blue P_1 at the 100th hour, for altering of strategic model structure becomes necessary if the winning probability does not conform with an enhanced or reduced strategic reconfigurability. The monotonic dependence of the Blue's winning probability on the specific and therefore strategic reconfigurabilities is clearly shown in Figure 1.5. The sensitivities of the dependence may depend on the parameter ranges selected.

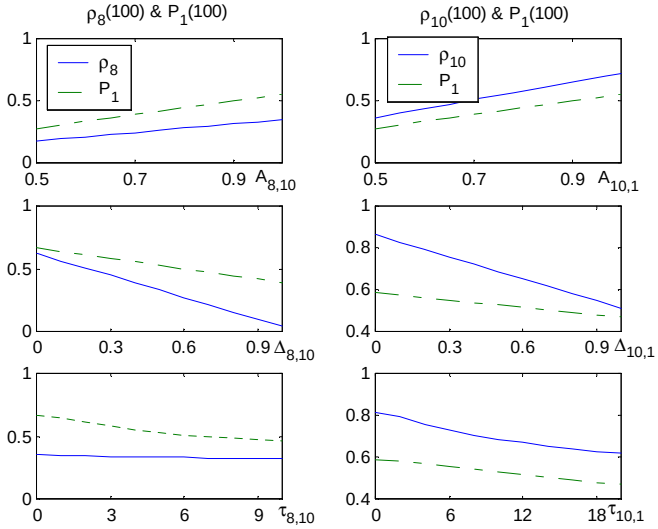


Figure 1.5 Specific reconfigurabilities ρ_8 (left) and ρ_{10} (right) v.s. transition coverage parameters ($A_{8,10}$, $\Delta_{8,10}$, $\tau_{8,10}$) and ($A_{10,1}$, $\Delta_{10,1}$, $\tau_{10,1}$), respectively.

1.4. Section summary

The section provided a way to exploit the potentiality and to understand the limitation of an air operation by scrutinizing the structure, and the dynamics within it through the notions strategic reconfigurability and transition coverage. These notions reflected Blue's ability to estimate the strategic state and to control the air operation. Although the aspects of modeling and analysis were emphasized, the results showed important implications on initial resource allocation, and on dynamical strategic planning and tactical execution for air operations.

Strategic reconfigurability has been defined as the arithmetic average of specific reconfigurabilities which

are essentially winning probabilities of Blue initiated at controllable states. The major distinction of this quantity from a general winning probability is that, besides being limited to the controllable strategic states, there is no a priori distribution attached to the strategic states. Therefore, contingencies that are controllable but less likely to occur will not be penalized. In fact from this viewpoint, one may modify Definition 2 by a set of criticality weighting factors $\{\alpha_x\}$, i.e.,

$$\rho = \frac{1}{|\mathcal{X}_t^c|} \sum_{x \in \mathcal{X}_t^c} \alpha_x \rho_x, \;\; 0 < \alpha_x \leq 1, \sum_{x \in \mathcal{X}_t^c} \alpha_x = 1.$$

As a final remark, we point out a numerical advantage of using the two-level air operation model. If large disparity exists among transition rates, the numerical stiffness can make the analysis of air operation difficult or impossible . This problem can be mitigated by absorbing the next state with fast transition to the originating state (assuming an infinitely fast transition), and capturing the dynamics of the fast transition in a tactical model.

2. Operational reconfigurability

2.1. Problem description

Our objective is to meet the demand of the ever decreasing cycle length in military air operations, which is the sum of the times required for planning, tasking, execution, and evaluation. This objective, when projected onto the expectation for a command and control (C2 hereafter) center, implies a more swift operation that involves information gathering, information processing, decision-making, and command issuing. The swiftness in turn requires a high level of C2 system availability. Availability of a system can be generally thought of as the fraction of the system uptime divided by the sum of the uptime and downtime. Our ultimate goal is to achieve nearly uninterrupted C2 operations.

It is apparent that a C2 center plays the role in an air operation as the controller in a feedback system. It carries out the functional mapping from information to decision in the feedback loop. A study conducted at the Draper Labs [14] that focuses on the effect of frequency of loop closure in air operations concludes that ability to close the loop at a higher rate (4 hour cycle v.s. 24 hour cycle), among other benefits, significantly shortens the time to achieve air campaign objectives. Other endeavors to enhance the C2 functional capability using different criteria and formalizations have also been reported [10], [11], [17], [41].

The underlying assumption so far has been that the structure which supports the functional mapping in a C2 center is always intact. In reality, however, a typical C2 center has grown to be a large and complex system, and this system is imperfect. Many subsystem failures can occur for many different reasons. For example, a miscarriage in information flow can be attributed to a broken link, a faded or jammed channel, a power outage, a failed sensor, an impaired storage device, a crashed processor, a human operator error, etc. In general, failures that disable the C2 functional mapping can be related to subsystems designated to perform data storage, transmission, processing, or interpretation. They impact information availability, integrity, and decision making in C2 centers. The current status in the effort to address these issues is still in the very early stage of installing monitoring tools. There is a severe lack of consideration in tackling the more fundamental issues of redundancy architecture and an appropriate level of automation for failure accommodation. The latter is important to mitigate unnecessary human errors and delays.

In our view, the concern over the loss of C2 system availability could be effectively addressed by a conscious effort of modification to the existing architecture to eliminate all single point failures. The term C2 system has been and will be used in the following development to represent the network of subsystems and components that host and support the functional mapping performed by the controller in a C2 center. The most efficient way to achieve the modification is to make use of and effectively manage the redundancy likely already in existence in the C2 systems. The rest of the section will explore such a possibility, and quantify the benefit of doing so to the overall air operation.

The section is organized as follows. In Subsection B, C2 system modeling is discussed for the purpose of availability analysis. The notion of operational reconfigurability is introduced to describe the effective level of redundancy. Subsection C discusses the assessment of C2 system availability under variable conditions such as subsystem failure rate, effectiveness of redundancy management, maintenance policy, and restoration rate. Subsection D discusses the effect of C2 system operational reconfigurability on the outcome of an air operation. Subsection E draws conclusions.

2.2. Operational reconfigurability

The first part of this subsection discusses qualitative availability modeling for a C2 system. Based on the model, the need for a measure on the effectiveness of redundancy management is argued, and the notion of operational reconfigurability introduced.

Availability [15] is the probability that a system is performing its required function at a given point in time when used under stated operating conditions. Among many definitions of availability, steady state availability will be considered, which represents the situation that the failure-restoration cycle has entered a steady state. Such a steady state definition will be assumed elsewhere in the section without further explanation. The availability value of a system is determined by the following factors

(i) reliability[1] distributions of individual subsystems and the functionalities of the subsystems in relation to the overall system;

(ii) the policy and capability by which the system is maintained[2], such as the decision on the restoration of failed subsystems and the distribution of the time required to do so;

(iii) the methods and the likelihood of success in management of existing redundancy, which are heavily influenced by our ability to monitor and diagnose subsystem failures, and to reconfigure the system upon identifying the failures.

Figure 2.1 shows a hybrid model [46] of an air operation. The discrete state strategic model at the top will be further explained in Subsection D. The tactical

[1]Reliability is the probability that a (sub)system will perform a required function for a given period of time when used under stated operating conditions

[2]Maintainability is the probability that a failed system will be restored to a specified condition within a period of time when maintenance is performed in accordance with prescribed procedures

model is represented in Figure 2.1 by a continuous state closed-loop control system which governs the execution of an air operation. The forward loop contains a model of battle dynamics. The tactical state vector may contain in its components, for example, the strengths of Blue assets, the rates of change of the strengths, their geographic locations, rates of change of locations, etc. The functional mapping carried out by the controller in a C2 center is represented by the two blocks in the feedback loop. It is responsible for generating two sets of signals. One is an estimate of the strategic state $\hat{x}$, and another is a corresponding control $u_{\hat{x}}$ to drive the tactical state ξ to wherever desirable. Availability of various subsystems in a C2 system is required in order to ensure that $\hat{x}$, and the parameters associated with $\gamma_{\hat{x}}$ and the estimator are available and correct, η and r are available and current, and finally $\hat{\xi}$ and $u_{\hat{x}}$ are current and correct.

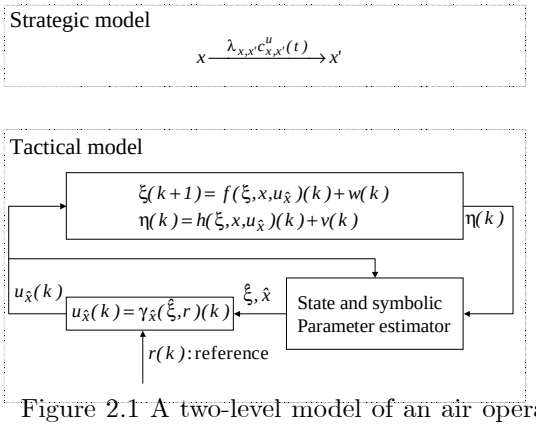


Figure 2.1 A two-level model of an air operation

An example of a functional decomposition of a C2 system is given in Figure 2.2 where the blocks marked TS (tactical and strategic sensors), DL (I/O control modules and data links), SM (storage media), CP (critical processors), and CS (critical software) represent some of the functional units. A functional unit is defined as a subsystem of a particular functionality that is necessarily available in order for the C2 system to be available. Each functional unit can be a complex interconnection of many subsystems. Considerable effort is usually necessary to arrive at a functional decomposition. Let A_{C2} denote the availability of the C2 system, and A_i is the availability of the ith functional unit. Then, the availability of a C2 system with N functional units is given by

$$A_{C2} = A_1 \times A_2 \times \cdots \times A_N. \tag{5}$$

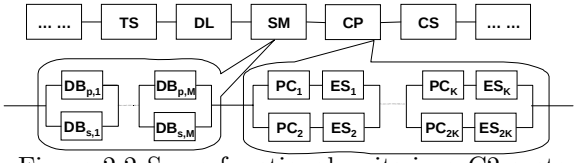


Figure 2.2 Some functional units in a C2 system

In the current C2 systems, vast opportunities exist for availability improvement without hardware addition, and without overburdening the subsystems in terms of processing speed, memory space, bandwidth, etc. The opportunities can be seized by, for example, assigning multiple tasks to multiple subsystems rather than assigning a single task to a dedicated subsystem within a functional unit, or using multiple copies of smaller data set for recursive processing rather than using a single copy of larger data set for batch processing. The expanded portions of Figure 2.2 show two examples of proposed architectural change for availability improvement. The original SM unit contains M storage media holding independent databases. These subsystems are named primary $\mathrm{DB}_{p,i}$ for $i = 1, \cdots, M$. Each primary subsystem is now appended with a redundant "cache", called a secondary $\mathrm{DB}_{s,i}$, using leftover storage space elsewhere to hold the most critical and immediately needed data. The original CP unit contains $2K$ processor cell-Ethernet switch pairs with non-overlapping tasks. Each pair is now equipped with the necessary (software) tools of one other pair, and a K series redundant CP unit is formed.

This work, however, is not intended to explore innovative ways to raise the level of redundancy, but to reason the significance and to assess the benefit of having an adequate redundancy level. A portion of the C2 system above will be used as a vehicle for our intended purpose. This portion is shown in Figure 2.3.

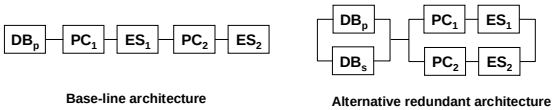


Figure 2.3 A glimpse of architectural change in C2

Most functional units within C2 are themselves complex interconnections of components. Statistical modeling [7] of the failure process of individual subsystems must follow carefully designed experiments of data collection, parameter (or distribution) estimation, and goodness-of-fit tests. Based on our initial investigation, failure rates (number of failures per unit time) of many individual subsystems are below 10^{-5}/hour, when intermittent failures are excluded. Therefore, subsystems are reliable. There is no doubt that intermittent failures will reduce the C2 availability. Modeling of intermittent failrues is our ongoing effort. In addition, both diagnosis and restoration for permanent failures of some subsystems can be lengthy processes (hours to tens of hours). The most fundamental reason for need of redundancy is the fragility of an architecture that allows single point failures. Individual subsystems do fail and can fail at an unfavorable time. The consequence to an air operation can be detrimental, as will be seen in subsection D. It is a fact to be kept in mind that statistical model development results from our lack of knowledge of the physical processes leading to a failure. As a consequence,

we can only infer from our sample of failure data to the general population, and our predictions tell little concerning an individual system or failure occurrence. Therefore, a non-redundant architecture with highly reliable subsystems is robust but fragile [9].

In order to measure the non-fragility. the notion of operational reconfigurability (OR hereafter) is introduced. In this section, OR is specific to characterize the C2 system survivability with respect to single point failures. Consider a canonical redundancy architecture with a parallel-to-series interconnection where each parallel interconnection in the outmost layer is considered a functional unit. The right side of Figure 2.3 shows a two-layer canonical interconnection. It is degenerated in the sense that there are no parallel interconnections in the inner layer. Suppose there are N functional units in a single layer canonical decomposition, and each has M_n $(n = 1, \cdots, N)$ subsystems. Let $c_{m,n}$ denote the coverage of the mth subsystem in the nth functional unit, where $c_{m,n}$ is define as Prob(n^{th} unit operates | its m^{th} subsystem has failed). Evaluating $c_{m,n}$'s is not a trivial task [43] because of its association with monitoring, diagnosis, and redundancy management policy. Its value also depends on how many remaining operating subsystems are in the functional unit. In general, the larger the number (M_n), the larger the value of $c_{m,n}$ due to reduced risk in redundancy management.

Operational reconfigurability OR for a single-layer parallel-to-series interconnected system is given by

$$OR \equiv \min_{n \in \{1,\cdots,N\}} \frac{1}{M_n} \sum_{m=1}^{M_n} c_{m,n}. \tag{6}$$

In a multi-layered parallel-to-series interconnection scheme, the expression would contain layers of minimum-average operations. OR points to the weakest functional unit in terms of its ability to manage the redundancy for covering its first failure. In particular, since $c_{m,n} = 0$ whenever $m_n = 1$, $OR = 0$ for any system that contains a non-redundant functional unit. This is a measure without the influence by a priori subsystem failure distributions, which is precisely needed to reflect the non-fragility.

In the representative C2 system of Figure 2.3, let OR_b denote the OR for the baseline architecture, and OR_r denote the OR for the alternative redundant architecture. Then $OR_b = 0$, and $OR_r = 1$, if $c_{m,n} = 1 \ \forall \ m, n$. In general, $0 \leq OR \leq 1$. OR essentially measures the available redundancy in a C2 system and how it is managed. Because of its dependence on coverage, it is reflective of monitoring and supervisory control performance. Such performance indicates the C2 ability to allow restoration of system function via reconfiguration upon subsystem failures. Reconfiguration can mean the removal of a failed subsystem, the switch-on of a spare subsystem, rescheduling of jobs, rerouting of the information flow, redistribution of the information storage, etc.

2.3. OR and availability

This subsection discusses availability modeling in a more quantitative manner. Its relation to OR, and other parameters such as restoration rate and maintenance policy, is of particular interest. Two simplifying assumptions are made here. (i) All subsystems in Figure 2.3 have exponential failure time distributions. (ii) All restoration time distributions are also exponential. Through out the subsection the representative C2 system of Figure 2.3 is used.

Viewed as a canonical form, the baseline architecture in Figure 2.3 contains only one type of functional unit. On the other hand, the alternative redundant architecture carries two types of functional units. The composite availability expression in (5) allows us to solve for the availability of individual units independently. A complete solution for availability requires the specification of the maintenance policy. The following policies are considered in our study:
(i) restoration to as good as new in one step with a prescribed restoration rate independent of the failure state when a (functional) unit level failure occurs;
(ii) restoration to as good as new in one step with the lowest restoration rate for the failure state when a unit level failure occurs;
(iii) restoration to as good as new in one step with a restoration rate determined by the sum of average restoration times of all failed subsystems associated with the failure state when a unit level failure occurs;
(iv) restoration to as good as new in multiple steps with a restoration rate determined by the criterion of quickest unit recovery or of the most important subsystem recovery (e.g., primary v.s. secondary) when a unit level failure occurs;
(v) restoration to as good as new in one step with a prescribed restoration rate independent of the failure state when any subsystem failure occurs;
(vi) restoration to as good as new in one step with the lowest restoration rate for the failure state when any subsystem failure occurs;
(vii) restoration to as good as new in one step with a restoration rate determined by the sum of average restoration times of all failed subsystems associated with the failure state when any subsystem failure occurs;
(viii) restoration to as good as new in multiple steps with a restoration rate determined by the criterion of most speedy unit recovery or of most important subsystem recovery when any subsystem failure occurs.

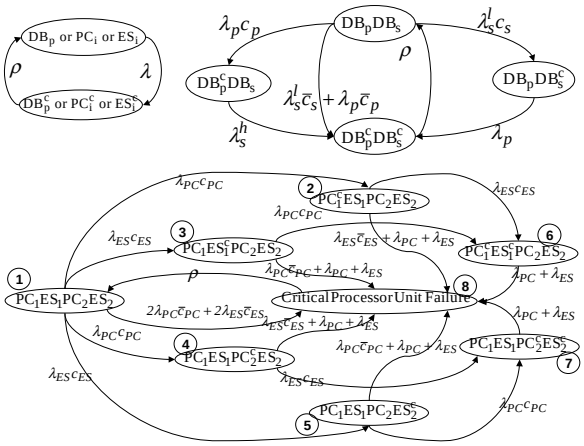


Figure 2.4 Rate transition diagrams of three types of functional units

The rate transition diagrams corresponding to the maintenance policy stated in (i) are shown in Figure 2.4 for the three types of functional units in Figure 2.3. Introduction of additional states is necessary under most of the other maintenance policies. The notations used in Figure 2.4 are as follows. λ denotes a failure rate, ρ denotes a restoration rate. A subsystem name appearing in a subsection of a state name indicates that the subsystem is up. A subsystem name with a superscript c appearing in a subsection of a state name indicates that the subsystem is down. c with appropriate subscript denotes coverage, and $\bar{c} = 1 - c$. Superscript l means a low value indicating, for example, that a subsystem is in standby mode, and superscript h means a high value (when the subsystem is no longer in standby). The state marked by "Critical Processor Unit Failure" has aggregated all CP unit failure states.

In general, for each of the three cases above, a set of state transition probabilities can be solved from the forward Kolmogorov equation $\dot{P}(t) = P(t)Q,\ P(0) = I$ which can be established directly from balancing the probability flow [8] from a rate diagram at each state. Therefore, transition rate matrix Q completely determines the set of transition probabilities. From the transition probabilities, any state probability can be easily calculated by setting appropriate initial state conditions. When the number of the states becomes large, numerical techniques and approximations must be sought to solve for the interested state probabilities directly from $\dot{\vec{p}}(t) = \vec{p}(t)Q, \vec{p}(0) = \vec{p}_0$, where $\vec{p}(t) = [p_1(t) \cdots p_n(t)]$ is the state (row) vector for an n-state functional unit.

Since our interest is in the steady state availability, the problem is much simplified. The steady state unavailability can be obtained by solving from the algebraic equation

$$\vec{p}_s Q_s = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \tag{7}$$

for steady state probability vector $\vec{p}_s$, where Q_s is obtained by replacing the state equation involving the derivative of unit failure state by $\sum_{i=1}^{n} p_i = 1$. Arranging the states for the redundant critical processor unit in the order as marked in Figure 2.3, and let $\lambda_c = \lambda_{PC}$, $\lambda_e = \lambda_{ES}$, $c = c_{ES} = c_{PC}$, $\rho = \lambda_{PC} + \lambda_{ES}$, and ρ^l (ρ^h) denotes a restoration rate.

$$Q_s = \begin{bmatrix} -2\rho & \lambda_c c & \lambda_e c & \lambda_c c & \lambda_e c & 0 & 0 & 1 \\ 0 & -\rho - \lambda_e & 0 & 0 & 0 & \lambda_e c & 0 & 1 \\ 0 & 0 & -\lambda_c - \rho & 0 & 0 & \lambda_c c & 0 & 1 \\ 0 & 0 & 0 & -\rho - \lambda_e & 0 & 0 & \lambda_e c & 1 \\ 0 & 0 & 0 & 0 & -\lambda_c - \rho & 0 & \lambda_e c & 1 \\ 0 & 0 & 0 & 0 & 0 & -\rho & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\rho & 1 \\ \rho^{l,h} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The unavailability of the functional unit is given by the state probability corresponding to the unit failure state. Figure 2.5 shows the results of evaluation the unavailability reduction factor,

$$URF \equiv \frac{1 - A_{C2}(OR_b)}{1 - A_{C2}(OR_r)} = \frac{1 - A_b}{1 - A_r}, \tag{8}$$

the ratio of the unavailability of the baseline to that of redundant architecture under maintenance policy (iii) for the SM unit (top), and for the CP unit (bottom), respectively. About a $20 \sim 95$ time reduction in unavailability in both units is observed for the range of failure rates indicated in Table O1, and for $\rho^l = 1/24$ hr^{-1}. Numerical comparisons are also made with respect to different maintenance polices listed above using two different restoration rates[3]. The results are not shown due to space limit.

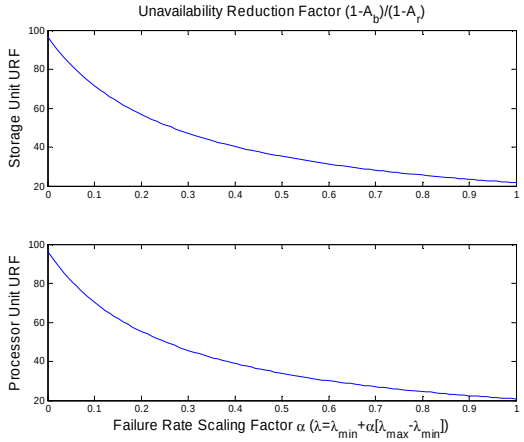


Figure 2.5 Unavailability reduction factor (URF) for SM and CP units due to redundancy architecture change

$\lambda_{PC} = \lambda_c$ (hr^{-1})	$9.0 \times 10^{-6} \sim 10^{-4}$	ρ^l (hr^{-1})	1/24
$\lambda_{ES} = \lambda_e$ (hr^{-1})	$7.4 \times 10^{-6} \sim 10^{-4}$	ρ^h (hr^{-1})	1/4
λ_p (hr^{-1})	$5.0 \times 10^{-6} \sim 10^{-4}$	OR_b	0
λ_s^l (hr^{-1})	$5.0 \times 10^{-7} \sim 10^{-5}$	OR_r	0.99
λ_s^h (hr^{-1})	$1.5 \times 10^{-5} \sim 10^{-3}$		

Table O1. SM and CP units failure, restoration rates, and OR numbers

[3]The authors would like to thank Ms. Xiaoxia Wang for her help in carrying out some of the availability calculations

2.4. OR and air operations

It is time to turn to the overall air operation, and investigate the benefit of a higher OR to the winning probability of Blue. This is an understandably difficult problem because of the conceivable complexity in establishing the linkage between the availability of the controller in a C2 center and the success of an air operation, though we have just established a definitive relationship between the availability and the OR. Fortunately, an earlier development [46] in our effort has provided the right framework to encourage an attempt. A brief review of the framework is in order.

A simple representation of a strategic model is shown at the top of Figure 2.1. It refers to the mathematical description of the evolution of a strategic plan in an air operation. It takes the form of a discrete state and continuous time Markov process. The model is specified by (i) a state space $\{\mathcal{X}\}$, (ii) a set of initial state probabilities $\{p_x(0), x \in \mathcal{X}\}$, and (iii) a set of state transition rates $\{\lambda_{x,x'}(t), x, x' \in \mathcal{X}\}$ from the current state x to the next state x' [8]. Figure 2.6 shows a low resolution example of a strategic model composed of 4 binary states: (Blue threatened, Blue defeated, Red targeted, Red defeated). A state of (True, False, True, False) can be represented by $x = 1010$ in binary. The meanings of the remaining states can be similarly explained. States 0000, 1000, and 1010 in Figure 2.6 are transient states and states 0100, 0101, and 0001 are absorbing states. Depending on whether preserving the Blue assets besides destroying the Red assets is also part of the mission of an air operation, the set of desirable outcomes for Blue can be one of $\{0101, 0001\}$ and $\{0001\}$.

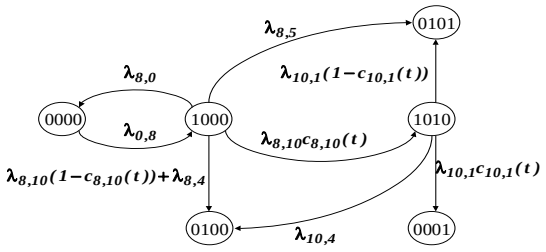


Figure 2.6 A low resolution strategic model

An important set of parameters introduced in our earlier work is the set of transition coverage values [46]. A transition coverage associated with a transition from x to x' is the conditional probability that the intended transition in fact occurs given that a triggering event has arrived. It is denoted by $c^u_{x,x'}(t)$, where u indicates its dependence on the control policy used in the tactical operation produced by the controller in a C2 center. It can be seen that a transition coverage serves to effectively modify the corresponding transition rate via $\lambda_{x,x'}c^u_{x,x'}(t)$. The transitions that have transition coverage attached to them are called controllable transitions. Blue's control objective is to maximize the transition coverage under the constraints of its resources and battle dynamics. The presence of C2 availability naturally modifies the originally defined transition coverage [46] for any controllable transition from x to x' in the following manner.

$$\mathcal{C}^u_{x,x'} = c^u_{x,x'}A_{C2},\ \bar{\mathcal{C}}^u_{x,x'} = c^u_{x,x'}(1 - A_{C2}) + (1 - c^u_{x,x'}) \quad (9)$$

where

$$\mathcal{C}^u_{x,x'} + \bar{\mathcal{C}}^u_{x,x'} = 1 \quad (10)$$

forms the Poisson decomposition [46] of the associated transition rate $\lambda_{x,x'}$. The original Poisson decomposition becomes a special case when C2 availability is perfect. It is obvious that the introduction of an imperfect C2 system availability reduces the effectiveness of the controller in the C2 center.

We now examine the average effect as well as the real-time effect of an imperfectly available C2 system on the air operation model of Figure 2.6. The following data are used in producing the result in Figure 2.7 and Table O2. $\lambda_{0,8} = 0.2$, $\lambda_{8,0} = 0.02$, $\lambda_{8,4} = 0.04$, $\lambda_{8,5} = 0.001$, $\lambda_{8,10} = 0.4$, $\lambda_{10,4} = 0.005$, $\lambda_{10,1} = 0.05$, $c_{8,10} = .95(1 - .5e^{-t/5})$, and $c_{10,1} = .95(1 - .5e^{-t/10})$. Modifications in transition coverage are as follows.

$$\mathcal{C}^u_{8,10}(t) = c^u_{8,10}A_{C2},\ \ \bar{\mathcal{C}}^u_{8,10} = c^u_{8,10}(1-A_{C2}) + (1-c^u_{8,10}),$$

$$\mathcal{C}^u_{10,1} = c^u_{10,1}A_{C2},\ \text{and}\ \bar{\mathcal{C}}^u_{10,1} = c^u_{10,1}(1-A_{C2}) + (1-c^u_{10,1}),$$

where various cases of A_{C2} considered are listed in Table O2. Function $1(t)$ in Table O2 denotes the unit step function. The 4 hour and the 24 hour time slots of unavailable C2 system correspond to the restoration rates used in the calculation of the previous subsection. The results are shown in Figure 2.7. Final winning

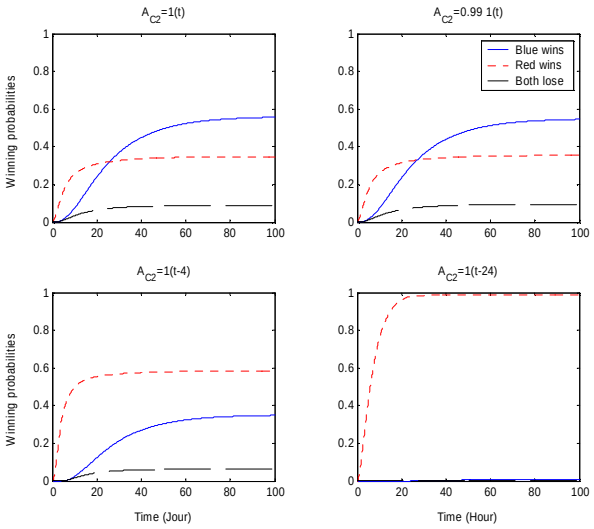


Figure 2.7 Winning probabilities up to the 100th hour of a military air operation

C2 availability	Blue wins	Red wins	Both lose
$A_{C2} = 1(t)$	0.56	0.35	0.09
$A_{C2} = 0.99 \times 1(t)$	0.54	0.36	0.10
$A_{C2} = 1(t-4)$	0.35	0.58	0.07
$A_{C2} = 1(t-24)$	0.01	0.99	0.00

Table O2 Winning probabilities at $t = 100$ hours when transient state probabilities have died out.

It can be seen from Figure 2.7 and Table O2 that a slight reduction in C2 availability has a limited effect on the outcome of the air operation on average. However, when the real-time unavailability of a C2 system falls within a critical period, the outcome can be disastrous. The latter case is shown in the two plots at bottom of Figure 2.7, and two items at the bottom of Table O2. These show where the fragility lies. An enhanced operational reconfigurability can reduce the unavailability and hence fragility by 2 orders of magnitude as shown in the example of the previous subsection. The reduction is achieved by filling in the periods of operation interruptions with a fairly unsophisticated usage of existing redundancy.

2.5. Section summary

This section delineated the importance and the potential of being able to provide and manipulate redundancy in the command and control system of a military air operation. The effort boils down to modification of the C2 system architecture so as to raise the system operational reconfigurability. An enhanced OR helps reduce the fragility of an otherwise robust system. The cost of reduction of fragility is the extra complexity of the system which must include diagnosis and management of redundancy (or supervisory control). The complexity introduced, however, is a miniature increment of a more costly and less carefully studied effort in setting up monitoring tools within C2 centers. Some simple but quantified case studies were presented to support our argument. Our ongoing effort is focused on more detailed availability modeling.

3. An example of supervisory control in C2

3.1. Problem desciption

In [45], a modern military air operation is modeled as a hybrid feedback control system. The impact of the redundancy architecture on the overall air operation is quantified, by which the functionality of the controller residing in a Command and Control Center (C2) is supported. The resilience of the architecture that reflects the quality of management of redundancy is measured by operational reconfigurability. The work concludes that successful use of redundancy is crucial in the reduction of fragility [9] in air operations in order to eliminate any single point failure in a C2 unit for which a long restoration period, relative to the mission time, is required.

Departing from [45] where all C2 units are modeled as static systems, this section considers a closed queuing network model [8] for the C2 processing unit depicted in Figure 3.1. As a consequence, less drastically different and more realistic outcomes are expected between uncontrolled and controlled operations when the unit experiences failures.

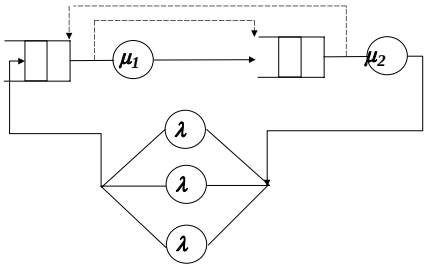


Figure 3.1 A C2 processing unit as a closed queuing network

The C2 processing unit under consideration contains two cells in series serving to complete two tasks in sequence for a single class (class 1) of arriving customers on a FCFS basis. There are 3 customers in the network. The status of a customer is elevated to class 2 as soon as it completes the service in cell 1, and the elevated status is removed as soon as the customer completes the service at cell 2. Each cell has a server with an exponential service time ($exp(\mu_i)$), an exponential life time ($exp(\nu_i)$), and an exponential restoration time ($exp(\gamma)$, to as good as new). The queue preceding each server has a sufficient storage capacity. A more complex model has been built for a more elaborate simulation study using Arena [23]. This complex model also contains a database unit and a human operator unit. The simulation results are reported separately in [48]. The delays captured in the three servers with exponential delay times ($exp(\lambda)$) in the feedback branch are intended to be also reflective of the response times of the ignored nodes in the simplified model. The simplification allows us to derive an analytic model for scrutinizing the effect of a supervisory control scheme on the processing unit.

Many analytical and numerical results on queuing systems with server failures can be found in the literature [1], [18], [40]. They are mostly too specific with regard to structural properties and operating policies to be widely applicable. Our goal here is to enhance the availability of the network without compromising the network performance such as the response time. To that end, a supervisory control scheme is to be attempted. Two control inputs are introduced: u_1, applicable to the first customer in cell 1, and u_2, applicable to the first customer in cell 2. Let Q_i, S_i, and C_i denote queue i, server i, and class i, respectively, where $i = 1, 2$.

$$u_1 = \begin{cases} 1, & \text{serve } C_1 \text{ at } S_1 \\ 0, & \text{serve } C_1 \text{ at } S_2 \text{ if } S_1 \text{ fails \& } S_2 \text{ idles} \end{cases}$$

$$u_2 = \begin{cases} 1, & \text{serve } C_2 \text{ at } S_2 \\ 0, & \text{serve } C_2 \text{ at } S_1 \text{ \& preempt-resume } Q_1 \text{ if } S_2 \text{ fails} \end{cases}$$

As a result, the operational reconfigurability (OR) for the network in Figure 3.1 is given by, according to the definition given in [45], $OR = 0$ without the supervisory control and $OR = 1$ with the supervisory control. This implies a perfect monitoring that aquires the state information in the network. Otherwise $OR < 1$.

In Subsection B, a Markov model for the processing unit in Figure 3.1 incorporating the above supervisory control is derived. In Ssubsection C, two measures of the unit's performance are evaluated with and without activating the supervisory control. One measure is the unit's steady-state response time to a customer (from entering cell 1 to existing cell 2), and the other is the steady-state availability which is roughly the fraction of time the network is up while the servers are not idled. Section IV discusses the effect of supervisory control on the overall air operation, and a possible extension of this work.

3.2. Markov model of the processing unit

With some abuse of the notations, the states in our Markov model are named $Q_1Q_2S_1S_2$, where $Q_i \in \{0, 1, 2, 3\}$ is the queue length at cell i with $Q_1+Q_2 \leq 3$, and $S_i \in \{0, 1\}$ represents server i intact (0), or failed (1). Figure 3.2 shows all the states in rectangular boxes and all the transition rates by the arcs linking the states. An alternative state name is marked by a circled number from $\{1, \cdots, 40\}$ near the corresponding state. The state transition rates are denoted by pairs in the form of a, b where a is the transition rate from a state named by smaller decimal number to a state named by a larger decimal number and b the rate back to the smaller

number.

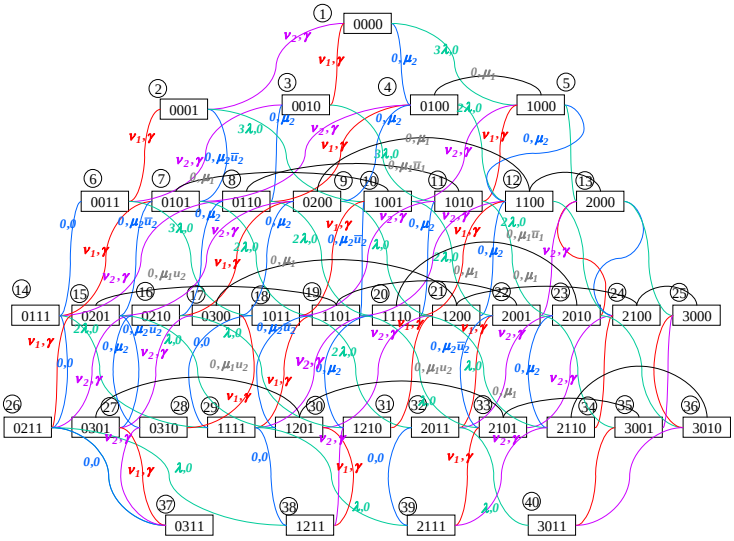


Figure 3.2 Rate transition diagram of the Markov model for the C2 processing unit in Figure 3.1

Denote the rate transition matrix by Q. Q is a 40×40 matrix. Only the nonzero entries involving control signals in Q are listed. These signify the exertion of supervisory control actions to state transitions in the way defined in Subsection A. In particular, $q_{7,2} = \mu_2(1-u_2)$, $q_{11,8} = \mu_1(1-u_1)$, $q_{15,7} = \mu_1 u_2$,$q_{19,15} = \mu_1 u_2$, $q_{19,10} = \mu_2(1-u_2)$, $q_{23,20} = \mu_1(1-u_1)$, $q_{27,15} = \mu_1(1-u_2)$, $q_{30,27} = \mu_1 u_2$, $q_{30,19} = \mu_2(1-u_2)$, $q_{33,30} = \mu_1 u_2$, $q_{33,22} = \mu_2(1-u_2)$, $q_{36,34} = \mu_1(1-u_1)$. Note that $\bar{u}_i = 1 - u_i$ in Figure 3.2.

In general, a set of state transition probabilities can be solved from the forward Kolmogorov equation $\dot{P}(t) = P(t)Q$, $P(0) = I$ which can be established directly from balancing the probability flow [8] from a rate diagram at each state. Since our interest is in the steady state probabilities, the problem is much simplified. Let p_i denotes the steady state probability for state i. Then $\vec{p} = [p_1 \ \cdots \ p_{40}]$ can be solved from

$$\vec{p}\,Q = \vec{0}$$

with one of its 40 scalar equations replaced by

$$\sum_{i=1}^{40} p_i = 1.$$

At $\lambda = 6$, $\mu = \mu_1 = \mu_2 = 12$, $\nu_1 = \nu_2 = 0.005$, $\gamma = 1/24$, for example, the set of steady-state probabilities shown in Figure 3.3 are obtained. All the quantities above carry the unit of a rate, i.e. $(time\ unit)^{-1}$. Many steady performance measures can be computed from these probabilities. Two of them will be detailed in the next subsection.

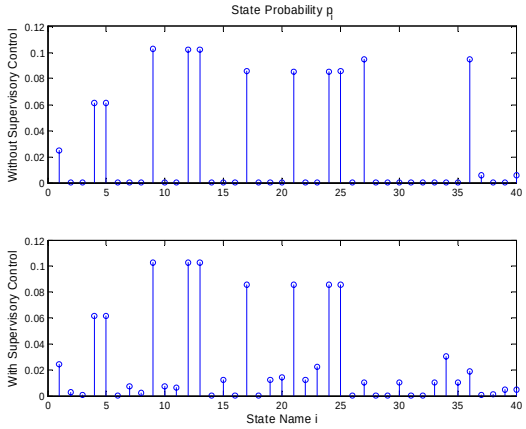


Figure 3.3 Steady-state prpbabilities without (upper) and and with (lower) supervisory control

3.3. Response time and availability comparison

The mean response time $E[T_R]$ can be solved for the processing unit by applying Little's Law to the top and the bottom parts of the queuing network in Figure 3.1 [8]. It has the expression

$$E[T_R] = \frac{3}{\mu(1 - Prob[S_2 \text{ not busy }])} - \frac{1}{\lambda}.$$

Let T_b be the response time of the processing unit without supervisory control, and T_r be the response time with supervisory control. Note that the subscripts are inherited from [45] where b stands for baseline and r stands for redundant. Define the response-time reduction factor as

$$RRF = \frac{T_b}{T_r}.$$

Figure 3.4 shows that RRF always improves with the use of supervisory control. RRF increases with increasing service rate, most prominently when delay time $1/\lambda$ is small, and RRF increases with increasing failure rate, most prominently when restoration rate is high.

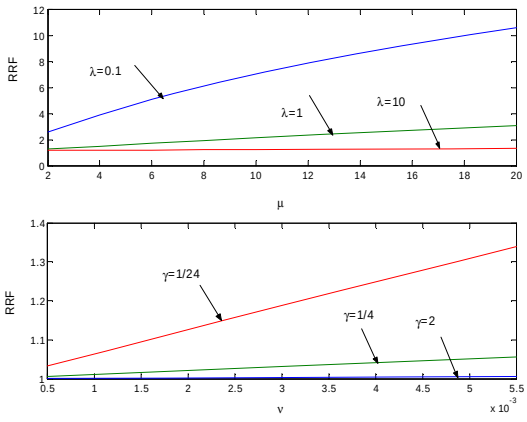


Figure 3.4 Response time reduction factor as functions of μ (upper) and ν (lower) with λ (upper) and γ (lower) as parameters

The unavailability of the processing unit is computed by the sum of state probabilities that contribute to the failure of the unit. For example, without the application of supervisory control, the unit unavailability is given by

$$U_b = p_7 + p_{11} + p_{14} + p_{15} + p_{18} + \cdots + p_{39} + p_{40}.$$

With the application of supervisory control the unit unavailability is given by

$$U_r = p_{14} + p_{18} + p_{26} + p_{29} + p_{32} + p_{37} + p_{38} + p_{39} + p_{40}.$$

Define the unavailability reduction factor as

$$URF = \frac{U_b}{U_r}.$$

Figure 3.5 shows that URF always improves with the use of supervisory control. URF generally decreases with increasing service rate, most prominently for some particular value of delay time $1/\lambda$, in this case 1 unit. URF always decreases with increasing failure rate ν.

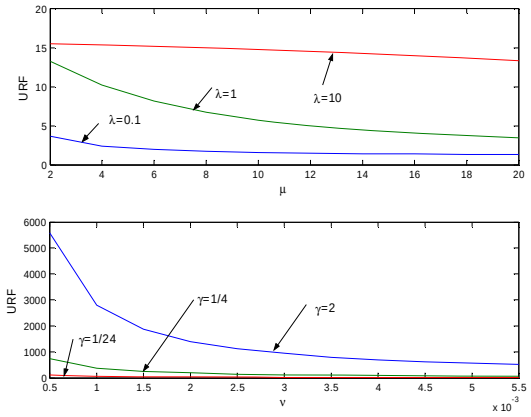


Figure 3.5 Unavailability reduction factor as functions of μ (upper) and ν (lower) with λ (upper) and γ (lower) as parameters

It is also observed that under the same structural parameters, URF is several times larger when the C2 processing unit is regarded as a static unit than that when it is regarded as a dynamic (queuing system) unit. This justifies our effort in queuing network modeling.

3.4. Section summary

The effect of the C2 availability on the winning probability of an air operation involving two opposing sides named Blue and Red (enemy) can be folded into the values of transition coverage of a high level strategic model [46] of the air operation. In particular,

$$\mathcal{C}^u_{x,x'} = c^u_{x,x'} A_{C2},\ \bar{\mathcal{C}}^u_{x,x'} = c^u_{x,x'}(1 - A_{C2}) + (1 - c^u_{x,x'}),$$

where A_{C2} is the C2 availability, $c^u_{x,x'}$ is the coverage for transition from strategic state x to a desirable strategic x' for Blue, u is the control action taken in the tactical air operation, and $\mathcal{C}^u_{x,x'} + \bar{\mathcal{C}}^u_{x,x'} = 1$. Therefore, the same conclusions as those drawn in [45] hold, which have been stated at the beginning of Subsection A.

The supervisory control applied to the processing unit in this is by no means the best possible control policy because it is set based on common sense rather than guided by a rigorous criterion. It is possible that criteria be formed by introducing costs associated with each control action or inaction. The policy derived under the criteria would be optimal in the sense that it minimizes some total cost index. This is called a Markov decision problem [8], and its solution to enhancing C2 reconfigurability will be pursued in the near future.

4. Fault-tolerance of multihop wireless networks (R)

4.1. Problem description

The class of wireless networks under consideration is the class of multiple-hop, distributed networks consisting of a large number of nodes. Each node has a limited energy supply that cannot be replenished, and is capable of packet transmission, reception, and processing that involves detection, fusion, coding and decoding. Our goal is to maximize the network reliability at its design life T_D[4].

Our main challenge is to develop a power covariate network reliability model[5]. As a result, the network reliability becomes the overarching measure that encompasses aspects of symbol error rate, energy efficiency, bandwidth efficiency, the effect of clustering, and the effect of feedback.

Many algorithms have been developed for the computation of node-pair reliability of networks, which is the probability that at least one route exists between a source node and a terminal node (Torrieri, 1994). Unlike any other networks, however, each route in our network itself forms a sub-network with an additional structure bound by the cooperative transmission scheme used. Therefore, we confine ourselves to the sub-network of a K-cluster route through which packets hop from cluster 1 to cluster K. The restriction to the single-route problem is entirely due to our intention to capitalize on some new physical layer transmission schemes (Li and Wu 2003; Li 2003, 2004). Our interest is not in devising routing protocols (Ordonez et al., 2004) that enhance the network connectivity evaluated using the knowledge of the spacial distribution of the wireless nodes (Xue and Kumar, 2004), or prolong network lifetime assessed using the deterministic knowledge of energy expenditure at each node (Bhardwaj et al., 2002). Instead, we are seeking to understand and to optimize the temporal evolution of network reliability and to utilize this information in the network operation with little supervising activity.

Existing schemes for enhancing the network fault-tolerance all carry significant overhead in terms of energy consumption. Examples of such schemes include multiple-path routing (Ganesan et al., 2002), packet replication (De et al., 2003), or feedback between neighboring nodes that either acknowledges a successful reception or requests a re-transmission of a packet (Kumar, 2001). Unlike more traditional networks, such as the Internet, where highly reliable links contribute little to the transmission failures, the links of wireless networks are much less reliable as a result of, for example, severe channel fading, or limited standalone reliability of low-cost nodes, or energy depletion of nodes. On the other hand, redundancy is abundant in such networks. Therefore, opportunities exist to address the issues of fault-tolerance and energy efficiency simultaneously. Of particular interest is the question on how much feedback is needed at a certain level of redundancy usage for a prescribed network design life.

With a proper formulation of a cooperative transmission problem employing multiple nodes, transmission diversity can be provided to combat deep-fading suffered by the near-ground communications (Laneman and Wornell, 2003; Sendonaris et al., 2003). The existing cooperative diversity schemes, though efficient in transmission power, increase the circuit energy consumption associated with, for example, static current in transceivers and encoding/decoding circuity, when multiple nodes must be kept on for listening and reception (Ganesan et al., 2002). We are developing new cooperative transmission schemes to address power efficiency, bandwidth efficiency, and fault-tolerance simultaneously. Our preliminary simulation results (Li and Wu, 2003) indicated a 6-fold reduction in power consumption at an enhanced level of network reliability with a two-node cluster that achieves a $15dB$ signal to noise ratio at the receiving cluster. This can be implemented using a new space-time block coding technique (Li, 2003 and 2004) with no loss of bandwidth efficiency.

Little has been discussed at the physical-layer in terms of network fault-tolerance (Hoblos et al., 2000) up to this point. Our basic idea is to determine the level of redundancy appropriate for our cooperative transmission scheme that also maximizes the network reliability at its design life. Since high cross-correlation among packets exists under this scheme, a certain packet loss rate could be tolerated without having to incur energy loss associated with frequent feedback and re-transmission.

The section is organized as follows. In Subsection B, a re-transmission chain is formed that serves to motivate the quest for understanding the impact of loop-closure on the network reliability. Subsection C discusses modeling the life time distribution of a node, and deriving the network level reliability and its lower bound as a function of link reliabilities. Subsection D applies the link reliabilities for the assignment of active nodes to clusters to maximize the network fault-tolerance up to its design life. It also tackles the re-transmission issue as a Markov decision problem with partial information feedback.

[4]A design life is defined as the maximum time by which a prescribed network reliability R^D is maintained, i.e.,$F^{net}(t)|_{t=T_D} = 1 - R^D$. where $F^{net}(t)$ is the cumulative distribution function of the network time to failure.

[5]The network reliability is given by $R^{net}(t) = 1 - F^{net}(t)$, which is defined as the probability that the network performs successfully its required function over a period of t time units under the stated operating conditions.

4.2. A motivating example

The Markov chain in Figure 4.1 describes a K-cluster packet transmission process where state name i stands for the ith cluster within which a packet hopping from the source through the network to the destination is residing. This chain is non-homogeneous due to the deteriorating link reliability p_i^l as the network ages. The link reliability, to be evaluated in the next subsection, is the probability that a packet reaching the i^{th} cluster is successfully relayed to the $i+1^{th}$ cluster with a required power level.

c_i, called a supervisory coverage, in Figure 4.1 is the conditional probability that upon the failure of the first transmission attempt, a re-transmission command is successfully issued to cluster i. In an unsupervised environment, $c_i = 1$ for the first transmission attempt, and $c_i = 0$ for any re-transmissions. In a supervised environment, on the other hand, $0 < c_i < 1$ in general (Wu, 2004). The factors affecting c_i include lack of observability of state, or erroneous state estimation, failure of a supervising node or cluster, fading channel linking the supervising cluster and cluster i, and collision among packets in which case a more elaborate queuing network model becomes appropriate. Therefore, in a truly distributed environment, it is reasonable to assume that $c_i \leq p_i^l$. $u(i)$in Figure 4.1 is the re-transmission control action when state i is entered. For the moment, $u(i) \equiv 1$ and time-invariant p_i^l are assumed.

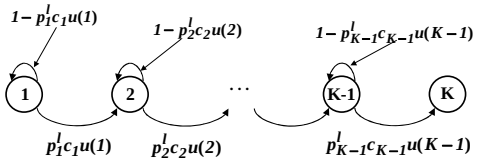


Figure 4.1 Packet transmission in a K-cluster route

With the Markov chain established, the state probability p_i^c, i.e., the probability that a packet is in cluster i, can be calculated by solving recursively for $\pi_i(k) = [p_1^c(k) \ \cdots \ p_K^c(k)]$ from $\pi_{k+1} = \pi_k \mathcal{P}_{k,k+1}$, where $\mathcal{P}_{k,k+1}$ transition probability matrix (Cassandras and Lafortune, 1999) as a function of $p_i^l c_i$.

Assume each transmission attempt consumes power P_i The average number of transmissions needed to reach state $i+1$ can be shown to be $\bar{N}_i = 1/p_i^l c_i, \quad i = 1, \cdots, K-1$. Then the power usage per packet transmission through the network, or power efficiency can be estimated by $\bar{P} = \sum_{i=1}^{K} p_i^c(\bar{N}_i P_i)$, and $E = \sum_{t=0}^{T_D} \bar{P}$ is an estimate of the network energy efficiency over its life time. Here the notion of the network age t is specialized to the number of packet transmissions that the network has carried out so far with the assumption all clusters age uniformly and the number of redundant nodes in each cluster is large.

Let us consider two simple but representative cases. In the first case there are no feedback and no supervisory activity, i.e., $c_i = 0$ for all re-transmissions, while the cluster transmission with multiple nodes is used. In the second case a supervisory scheme is in place to issue re-transmission whenever needed, while only a single node in a cluster is used at a time for each transmission attempt.

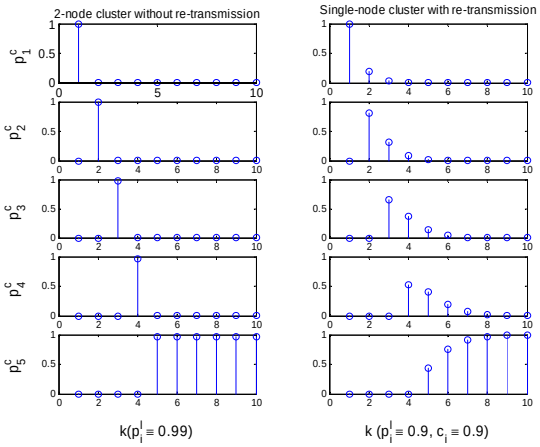


Figure 4.2 2-node w/o feedback v.s. 1-node w/ feedback

Figure 4.2 shows 10 snapshots of state probabilities for a 5-cluster route when a packet transmission is initiated at $k = 1$ for the above mentioned two representative cases. The five rows of the plots are p_1^c through p_5^c at 10 consecutive instants of packet transmissions. The left column of plots is for the unsupervised case, where the link reliability $p_i^l \equiv 0.99$ for all i at the current network age, resulting from a 2-node cooperative transmission with a reliability of 0.9 for each node. For the moment perfect channels are assumed, in which case a link reliability is the same as a cluster reliability. The right column of plots is for the supervised case, where the current link reliability $p_i^l \equiv 0.9$ for all i, resulting from a 1-node/transmission scheme with a node reliability also 0.9, and a supervisory coverage $c_i = 0.9$.

The following can be observed. (i) Without feedback, the network reliability $\prod_{i=1}^{K-1} p_i^l$ depends solely on the individual link reliabilities. Therefore, high link reliability is crucial, especially for a route with a large number of hops. Given the limited standalone node reliability and channel fading phenomena, high link reliability is not possible without using a multiple-node cooperative transmission scheme. (ii) Feedback enables the network to eventually settle in its absorbing state at the expense of power and bandwidth expenditures. More specifically, it takes an average of 1.23 transmissions to send a packet to the next cluster in this example, which leads to less power efficiency, and more delay. In conclusion, it is most desirable to have a supervisory scheme that is, however, rarely called for under high coverage and high link reliability conditions.

Our remaining tasks have become obvious: to assess and maximize link reliabilities, and to devise a re-transmission stopping rule that abandons a route when

it becomes a liability to the network.

4.3. Network reliability

4.3.1. Node and channel reliability models: Due to dependence on power consumption, time to failure distribution of a node must be of increasing failure rate (IFR), i.e., a node that is found to be good after some usage must have a shorter residual life than a brand new node. Weibull IFR distribution

$$F^n(t) = 1 - r^n(t) = 1 - e^{-\left(\frac{t}{\theta(P(J))}\right)^{\beta(P(J))}} \tag{11}$$

is used as an example in this section, where $\beta(P(J)) > 1$ is called a shape parameter and $\theta(P(J)) > 0$ is called a characteristic life. The Weibull model is deemed covariate because of its explicit dependence of the parameters on power $P(J)$ joules/packet/node involving an J-node cooperative transmission. For simplicity $P(J)$ will be suppressed in the following discussion. t is now the identified with the number of packets the node has relayed. The characteristic life can be scaled by $1/\bar{N}_i$ to reflect the additional life expenditure due to the need of re-transmission at the i^{th} cluster.

For a given type of node and a family of distributions, the parameters of the distribution can be determined statistically (Casella,2002). Suppose at a fixed power level, an n-unit concurrent test is performed. The test terminates at the arrival the rth node failure, i.e., upon the observation of failure times $\{t_1, \cdots, t_r\}$. The maximum likelihood estimates of the Weibull parameters can be solved from

$$\frac{n}{\hat{\beta}} + \sum_{i=1}^{r} \log t_i - \frac{1}{\hat{\theta}} \sum_{i=1}^{r} t_i^{\hat{\beta}} \log t_i + (n-r) t_r^{\hat{\beta}} \log t_r = 0$$

$$\frac{n}{\hat{\beta}} + \frac{1}{\hat{\theta}^2} \sum_{i=1}^{r} t_i^{\hat{\beta}} + (n-r) t_r^{\hat{\beta}} = 0.$$

In addition, Mann's two-parameter F-test can be performed to determine whether to reject the hypothesized Weibull with a specified significance level (Zacks,1992). The empirical dependence of β and θ on $P(J)$ can be established by repeating the experiments for many power levels.

Let T_{lc} denote the period of loop closure, indicating how often a node is checked out to determine whether it has failed. Assuming a uniform aging process, the residual life distribution $F_k(t) \equiv P[T \le t | T > (k-1)T_{lc}]$ of a node follows

$$F_k(t) = 1 - \frac{r_i^n(t)}{r_i^n((k-1)T_{lc})}, \;\; t \ge (k-1)T_{lc}, \;\; k = 1, 2, \cdots.$$

Single channel failure distribution is assumed to be time independent, and identical for all channels in the network, i.e., $r_i^c = r^c$, unless some a priori information is available, which can be easily incorporated. The randomness is associated with the fading phenomena (Rappaport,2002).

4.3.2. Link and network reliability: Suppose the i^{th} cluster of the K-cluster network contains a total of I_i nodes. Suppose for every sequence of I_i requests of packet transmission that arrive at the i^{th} cluster, a node responds to a set of J_i consecutive requests. In such an arrangement which will be called a participating/non-participating protocol hereafter, the burden of packet transmission for every node is effectively reduced to a fraction J_i/I_i, and the single node characteristic life θ_i is increased effectively to $\theta_i I_i/J_i$. Note again that the current age of a node is the number of packet transmissions the node has carried out so far. This protocol unifies the node ages across a cluster.

The reliability of the K-cluster network is now considered. The example in Figure 4.3(a) depicts a portion of an interconnection containing two nodes in each cluster, where S_j^i denotes the j^{th} node in the i^{th} cluster, and $C_{j,k}^i$ denotes the channel linking the j^{th} node in the i^{th} cluster to the k^{th} node in the $i+1^{th}$ cluster. The consideration of channel failures turns the interconnection into a nested structure rather than a cascade structure.

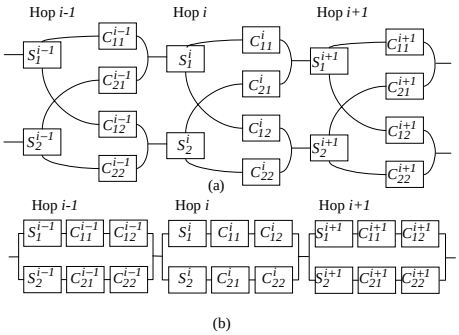


Figure 4.3 (a) Dependence diagram, (b) simplification

The nested structure in Figure 4.3a can be decomposed into logic stages for which the output signal availability can be computed when conditioned on the input signal availability using a combinatorial method. More specifically, one may write for the ith hop in Figure 4.3(a)

$$y_1^i = C_{11}^i S_1^i u_1^i + C_{21}^i S_2^i u_2^i, \;\; y_2^i = C_{12}^i S_1^i u_1^i + C_{22}^i S_2^i u_2^i,$$

for which sixteen conditional probabilities

$$P(y_1^i y_2^i = ab | u_1^i u_2^i = cd), \;\; a, b, c, d \in \{0, 1\}$$

can be computed with a '1' representing availability of a signal and a '0' unavailability. For example, with t suppressed, it can be shown that

$$\begin{array}{l} P(y_1^i y_2^i = 00 | u_1^i u_2^i = 11) \\ = (1 - r_i^n)^2 + 2(1 - r^c)^2 r_i^n (1 - r_i^n) + (1 - r^c)^4 (r_i^n)^2 \\ P(y_1^i y_2^i = 01 | u_1^i u_2^i = 11) = P(y_1^i y_2^i = 10 | u_1^i u_2^i = 11) \\ = 2r^c(1 - r^c) r_i^n (1 - r_i^n) + (2r_c - (r^c)^2)(1 - r^c)^2 (r_i^n)^2 \\ P(y_1^i y_2^i = 11 | u_1^i u_2^i = 11) \\ = 2(r^c)^2 r_i^n (1 - r_i^n) + (r^c)^2 (1 - r^c)^2 (r_i^n)^2 \end{array} \tag{12}$$

The stages are linked by $u_1^{i+1} = y_1^i$, $u_1^i = y_1^{i-1}$, $u_2^{i+1} = y_2^i$, and $u_2^i = y_2^{i-1}$.

Extension of the above result from a 2-node clusters to a J_i-node cluster is straightforward, and can be carried out in a systematic manner. Nevertheless, reliability evaluation of the nested structure is a major hurdle for optimization, especially in real-time. It is therefore desirable to work with simpler network reliability models that provide bounds on the nested network reliability. For example, with a k_i-out-of-J_i (Zacks,1992) requirement based on cooperative transmission considerations, where k_i is the required minimal number of operative nodes and J_i is the number participating nodes in the ith cluster, R^{net} is bounded below by

$$\prod_{i=1}^{K} \sum_{r=k_i}^{J_i} \begin{pmatrix} J_i \\ r \end{pmatrix} [(r^c)^{J_{i+1}} r_i^n]^r [1-(r^c)^{J_{i+1}} r_i^n]^{J_i - r}, \quad (13)$$

which comes from the decomposed cascade of functional units as shown in Figure 4.3(b). Note that $J_{K+1} = 0$ because no further transmission is needed at cluster K. The lower bound is equivalent to the configuration of Figure 4.3(a) in that signals initiated from node S_j^i can reach every participating node in hop $i+1$ if and only if every channel C_{jk}^i is intact for the given i, j, and for all $k \in \{1, 2, \cdots, J_{i+1}\}$. This implies a channel reliability $r_c^{J_{i+1}}$. It is, however, not necessary that every channel must work to guarantee the information flow through the network, hence the conservativeness. Let $R_i^{J_j}$ be the probability that a packet reaches at least k_{i+1} nodes among the J_{i+1} participating nodes in cluster $i+1$ with the required power level, given that the packet is transmitted at cluster i from at least k_i nodes among J_i participating nodes. Denote by the i^{th} term in the product in (13) as $\underline{R}_i^{J_i}$.It can be shown that $0 < R_i^{J_j} - \underline{R}_i^{J_i} < 1-(r^c)^{J_{i+1}}$. The error bound is tight as long as channel reliability is high, and the number of participating node in cooperative transmissions is not excessively large. Many of the analyses from this point on will use the lower bound (13), including the definition of link reliability, i.e., $p_i^l \equiv \underline{R}_i^{J_i}$, and composite network reliability $\underline{R}^{net} = p_1^l \times \cdots \times p_K^l$. Now, the participating node allocation problem becomes amendable to solutions using dynamic programming (Bellman, 1957).

4.4. Optimization and control

This subsection discusses two applications of the derived link reliabilities.

4.4.1. Participating node allocation: Our task is to determine the values of J_1, J_2, $\cdots$, J_K so that the network reliability is the largest at T_D without violating a bandwidth constraint. In cluster i, $J_{i,min}$ is imposed by the particular transmission scheme, while $J_{i,max} \le I_i$ is mainly imposed by the available bandwidth. Bounding model (13) converts the network level decision into a series of coupled cluster level decisions. In this case, channel failures introduce only local coupling which can be resolved by an ordered selection process starting from J_K at the last cluster and ending at J_1. The solution $\{J_1^*, \cdots, J_K^*\}$ can then be inserted to the staged conditional probability formulae (12) to calculate the true network reliability.

To illustrate the basic idea, consider a 3-cluster network with 10 nodes in each cluster. A tree structure shown in Figure 4.4 can be created to represent all possible solutions at T_D, where all branches violating the constraints have been trimmed. Constrains particular to the cooperative transmission scheme (Li and Wu, 2003) are $\sum_{i=1}^{3} J_i \le 12$ and $J_{i,min} = 2$. Each joint of the tree at a given cluster index represents a possible cumulative number of nodes. Each branch leading to the joint carries a cost equal to $\underline{R}_i^{J_i}(T_D)$ for a particular J_i. The accumulated reliability for each passage from the root to a leaf can be computed using Bellman's principle of optimality (Bellman, 1957). The principle is applied at every unit index i by comparing all the accumulated reliabilities leading to the same joint. Only the solution of the highest reliability is retained at each joint, and the rest are removed. Once the set $\{J_1^*, J_2^*, \cdots, J_K^*\}$ is obtained, the link reliabilities are set to $p_i^l = \underline{R}_i^{J_i^*}, i = 1, 2, \cdots, K-1$. Suppose unit reliabilities $\underline{R}_i^2(T_D)$ through $\underline{R}_i^8(T_D)$ have been found to be 0.85, 0.90, 0.95, 0.99, 0.995, 0.999, and 0.9995, respectively, for the network in Figure 4.4, the optimal node allocation derived using dynamic programming is: $J_i = 4$ for $i = 1, 2, 3$.

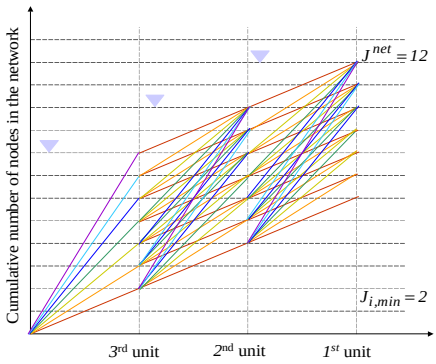


Figure 4.4 Trellis diagram for node allocation

Note that unit reliability is a complex function of J_i, which is determined by the methods discussed in subsections 3.1 and 3.2. The optimization in this subsection is carried out under the assumption that network is operating unsupervised. It is possible to re-optimize the network reliability projected at the network design life when supervisory exists that can report the actual rather than the predicted status of the nodes. A commonly used idea called a receding horizon optimal control in the control literature (Mayne, 1990) can be applied in this case. Though only limited data

exchange is required to carry out dynamic programing, the main challenge with real time optimization in a distributed environment is that data exchange is not only expensive but unreliable. How frequently such a partial reorganization should be performed is currently under investigation.

4.4.2. Re-transmission control: In this subsection, the re-transmission chain of subsection 2 is revisited. It is now assumed that the network is supervised to the extent that it can detect a cluster transmission failure but not the state of the nodes and channels, and the participating/nonparticipating protocol is effective to manage the large number of nodes available at each cluster. The decision regarding re-transmission in each of the clusters upon the detection of a cluster transmission failure can be made based on the solution of a Markov decision problem. The main purpose is to be able to terminate the service of the K-cluster route so that it does not turn into a black hole in the network.

Let $X_k \in \{1, 2, \cdots, K\}$ denote the random state variable at $t = k$ in the chain. Control action $u(x_k) = 1$ (or 0) indicates the network's decision to (or not to) re-transmit a packet. Let $C(x_k, u_k)$ be the cost incurred when control action u_k is taken based on x_k. Our goal is to determine a re-transmission policy π that minimizes the total expected cost $V_\pi(x_k = i) = E_\pi \sum_{k=0}^{\infty} C(X_k, u_k)$. It has been shown that under the condition $0 \leq C(j, u) < \infty$ for all j and all u that belongs to some finite admissible sets U_j, $j = 1, 2, \cdots, K-1$, the minimal cost $V^*(i)$ satisfies the following optimality equation (Cassandras and Lafortune, 1999)

$$V(i) = min_{u \in U_i}\{C(i, u) + \sum_{j=1}^{K} p_{i,j} V(j)\}.$$

In addition, policy π^* is optimal if and only if it yields $V^*(i)$ for all i.

Referring to the Markov chain in Figure 4.1, the optimality equation can be specialized to the following form.

$$\begin{aligned} V(i) = min_{u \in U_i}\{&\underbrace{u(i)T_i + [1 - u(i)]L_i}_{C(i,u(i))} + \\ &+u\underbrace{[p_{i,i}(u(i))V(i) + p_{i,i+1}(u(i))V(i+1)]}_{\sum_{j=1}^{K} p_{i,j} V(j)}\} \end{aligned} \tag{14}$$

where $p_{i,i} = 1 - p_i^l c_i$, $p_{i,i+1} = p_i^l c_i$, where network age t is suppressed, T_i is the power and bandwidth cost incurred when the network chooses to re-transmit a packet, and L_i is the packet loss cost incurred when the network chooses not to re-transmit. (14) can be expressed as

$$V(i) = min\{T_i + p_{i,i} V(i) + p_{i,i+1} V(i+1), L_i\}$$

To gain some insight into the optimal policy, assume $T_i = T$, $L_i = L$, $p_{i,i+1} = r$, and $p_{i,i} = 1 - r$ for $i = 1, \cdots, K-1$. Since

$$(1-r)V(j) + rV(j+1) < (1-r)V(i) + rV(i+1)$$

as long as $j > i$, the optimal policy is of the threshold type (Cassandras and Lafortune, 1999) with some threshold i^*, i.e.,

$$V(i) = \begin{cases} T/r + V(i+1), & i > i^*, \quad (u(i) = 1) \\ L, & i \leq i^*, \quad (u(i) = 0) \end{cases}.$$

Given that $V(K) = 0$, $V(i)$ can be solved

$$V(i) = \begin{cases} (K-i)T/r, & i > i^*, \quad (u(i) = 1) \\ L, & i \leq i^*, \quad (u(i) = 0) \end{cases},$$

from which the threshold is obtained

$$i^* = \lceil K - \frac{rL}{T} \rceil.$$

$\lceil . \rceil$ denotes the smallest nonnegative integer greater than $K - rL/T$. It can be seen that the optimal policy favors a re-transmission when a packet is near the end of the K-cluster route (large i), when a cluster is young (large r), when the cost of a packet loss is large (large L), when power & bandwidth are cheap (small T), when a route is short. (small K). A study without the simplifying assumptions along this direction is ongoing.

4.5. Section summary

In this section, fault-tolerance of a K-hop wireless network with a cooperative transmission scheme is measured by the network reliability projected at its design life. The network is subject to both channel fadings and node failures. A node life time is modeled by a Weibull-like covariate model where both the characteristic life and the shape parameters are dictated by the node power consumption. Assessment of network reliability is accomplished by the composition of staged conditional probabilities. Upper and lower bounds on the network reliability are obtained that disentangle the nested interconnection of nodes and channels into a chain of links with decoupled link reliabilities.

An analysis framework has been established in this section for qualifying fault-tolerance in wireless networks that age more rapidly as its node power is being consumed. Improving fault-tolerance is shown to require a high supervisory coverage with a frequency of loop closure as low as affordable by the high link reliability. A more thorough investigation is ongoing on the effect of the loop closure frequency on the network fault-tolerance.

5. Effect Of Acknowledgement On Performance Of A Fault-Tolerant Wireless Network

5.1. Problem desciption

This section studies the same *K*-hop, single-route wireless network, shown in Figure V.1, as that considered in [47], however with several realistic constraints included. The objective is to quantify the effect of loop closure frequency and the nodes' storage capacity on the performance of the network in terms of network lifetime, and packet loss rate. The constraints considered are node's finite packet processing time, node's life expenditure while performing supervisory activities, and a specific re-transmission protocol. As a result of the added complexity, it becomes necessary to resort to numerical means in order to achieve our objective. A discrete event simulation tool called Arena [35] is used for this purpose.

Background information summarizing the conditions

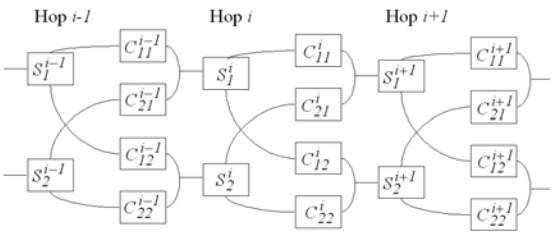


Figure 5.1 Dependence diagram of 3 clusters of nodes and channels in a K-cluster wireless network

The example in Figure 5.1 depicts a portion of an interconnection containing two nodes in each cluster, where S_j^i denotes the j^{th} node in the i^{th} cluster, and C_{jk}^i denotes the channel linking the j^{th} node in the i^{th} cluster to the k^{th} node in the $i+1^{\text{th}}$ cluster. Modeling channel failures turns the interconnection into a nested structure rather than a cascade structure. This model is used in [47] to understand the effect of the level of cooperation in transmission on network reliability at its design life[5], as well as the benefit and cost of feedback.

Each node in Figure 5.1 has a limited energy supply that cannot be replenished, and is capable of transmitting and receiving symbols in packets, and processing signals, which involves detection and fusion, coding and decoding, modulation and demodulation, as well as channel estimation. The restriction to the single-route problem is entirely due to the intention to capitalize on some new physical layer transmission schemes [30], [28], [29] rather than on the routing protocols.

[5] network's design life is defined as the maximum time by which a prescribed reliability of the network is maintained.

In [47], lifetimes of the nodes are modeled with power-covariate distributions of increasing failure rates. A method for assessing both link and network reliabilities projected at the network's design life is developed. The link reliability is then used to allocate active nodes to clusters through dynamic programming to maximize the network's fault-tolerance, and to establish a re-transmission control policy that minimizes the expected cost involving power, bandwidth expenditures, and packet loss.

A Markov chain model is established in [47], as shown in Figure 5.2, to capture the high-level effect of feedback, where state name *i* stands for the i^{th} cluster within which a packet hopping from the source through the network to destination is residing.

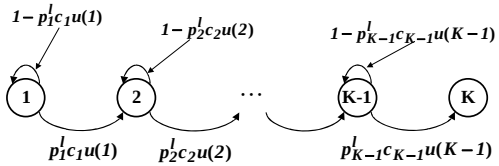


Figure 5.2 Packet transmission in a K-cluster route

c_i, called a supervisory coverage, is a conditional probability that upon the failure of a first transmission attempt to the $i+1^{\text{th}}$ cluster, a re-transmission command is successfully issued to cluster *i*. The factors affecting c_i include lack of state observability, fading channel from the receiving node/cluster to transmitting node/cluster. It captures decision risks in supervisory activities. $u(i)$ is the re-transmission control action when state *i* is entered. A re-transmission is attempted when $u(i) = 1$. This chain is non-homogeneous due to the dependence of link reliability p_i^l on cluster age. The link probability is the probability that a packet reaching the i^{th} cluster is successfully relayed to the $i+1^{\text{th}}$ cluster with a required power level, and depends on both channel and node reliabilities.

The following can be observed from the Markov model. (i) Without supervisory activities, the network reliability depends solely on the product of all link reliabilities. Therefore, high link reliability is crucial, especially for a route with a large number of hops. Given the limited standalone node reliability and channel fading phenomena, high link reliability is not possible without using clustered cooperative transmission scheme. (ii) Feedback enables the network to eventually reach the destination at the expense of power and bandwidth expenditures, and time delays.

To determine whether to retransmit in case of a transmission failure, [47] assumes that the network is supervised to the extent that it can detect a cluster transmission failure but not the state of the nodes and channels. The decision regarding re-transmission in each of the clusters upon the detection of a cluster transmission failure is made based on the solution of

a Markov decision problem. The main purpose is to be able to terminate the service of the *K*-cluster route so that it does not turn into a black hole blocking the traffic of other routes that intersect it.

This section will investigate the dependence of network lifetime and packet loss rate on the frequency of acknowledgements from the receiving nodes, and on the storage capacity of the nodes for both arriving packets and the copies of transmitted packets in case of a transmission failure. The purpose of acknowledgement is to terminate the activity of the *K*-cluster route so that the lives of operative nodes are saved for other usages. No re-transmission of packets is considered in this section. On the other hand, the following simplifying assumptions used in [47] are eliminated in this section: processing time of packets within a node is zero; supervisory activities incur no power expenditures; supervisory scheme has no association with any particular protocols. Elimination of these simplifying assumptions complicates our tasks dramatically that it forces us to resort to numerical means for performance evaluation.

The section is organized as follows. Subsection B describes a particular feedback protocol implemented in Arena for the next cluster to acknowledge the receipt of a certain number of packets. The section also highlights modeling of the *5*-hop network of Figure 5.1 with Arena. Subsection C defines a set of performance measures that include network lifetime, packet loss rate, and false alarm. It then details the analyses of performance based on the simulation output. Subsection D discusses the implications on network design based on our simulation study, limitations of our work, and areas to be addressed in the future.

5.2. Acknowledgement protocol and network modeling with ARENA

Referring to Figure 5.1, each receiving node at the next cluster transmits an acknowledgement (*ACK*) to the transmitting cluster in the previous cluster after receiving a sequence of *N* packets; this is regarded as a feedback. Therefore, the loop closure period is *N*. If, for example, a transmitting node, after transmitting a string of *DL* packets, where *DL>N*, does not receive an acknowledgement (*ACK*) from the receiving node, it assumes that all receiving nodes in the cluster have failed. In this event, the transmitting simply stops sending packets, and the network life ends. The supervisory coverage [42] in this case is the conditional probability that upon the reception of *N* packets, an *ACK* command issued by one of the receiving nodes successfully reaches one of the working transmitting nodes. In a network with redundant nodes, it is not necessary that every channel must work to guarantee the information flow through the network.

In spite of an analytic approach proposed in [47], the evaluation of network reliability for the nested structure is tedious and is a major hurdle for optimization. The additional constraints described in the previous subsection completely rules out the possibility of analytical approach for performance analysis. For this reason, the model in Figure 5.1 is constructed with Arena [35]. Arena is a general-purpose simulation tool of discrete event systems. Though it does not have the network-oriented convenience afforded by specialized tools for networks, it offers the flexibility for us to model in detail many aspects of networks that have not been studied by others.

The model of the wireless network shown in Figure 5.3 is constructed using different modules in Arena that are arranged into a number of templates such as 'Basic Process', 'Advanced Process' and 'Advanced Transfer' [23]. The Basic Process template contains modules that are used in modeling packet arrival and packet departure, assigning attributes to packets, channel random fading, and node processing. The Advanced Process panel comprises specific logical functions such as a fictitious control logic unit that is used to match the incoming packets depending on their attributes, duplicate, and merge packets. Finally, the Route module in Advanced Transfer template is used to transfer the packets to specified stations. Independent replications are performed for each simulation of the wireless system model and the simulation results are stored and reported.

In the *5*-hop wireless network of Figure 5.3, the data packets are generated at the source of the network with a Poisson rate. They pass through channels and nodes, and are delivered to sink. For simplicity, assume that the source and sink do not fail over time. There is no transport time for passing through channels. The processing time at each node is fixed at *T* units of time per packet. A packet can be lost through a faded channel, in a failed receiving node, in a failed transmitting node, or due to a collision. The channels have independent failure probabilities. The nodes have failure times that follow independent Weibull distributions. Weibull distribution allows a more truthful description of a node's life in our application where a node that is found to be good after some usage will have a shorter residual life than a brand new node, while the used node found to be good is indistinguishable from a new one if it is modelled by an exponential lifetime distribution.

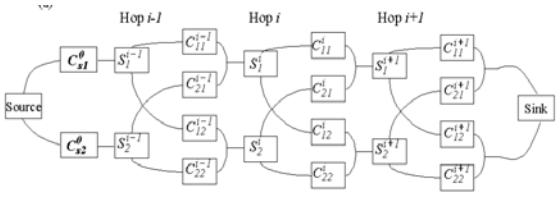


Figure 5.3 A *5*-hop wireless network

The major source of randomness for the wireless network is linked to node failures. Suppose S_{B1}, S_{B2}, … are the busy time data of a node and F_{S_B} is the distribution fitted for these data. Then the node is working until its total

accumulated busy (processing) times reaches a value s_{b1}, at which point the busy node fails.

Each receiving node at the next hop transmits an acknowledgement (*ACK*) to a transmitting node of the previous hop after receiving a sequence of *N* packets. *N* is referred to as the loop closure period. This transmission of acknowledgement to the previous node acts as a feedback. The transmitting node of the previous cluster inhibits the packet transmission to the next sensor if it does not receive the acknowledgement within a certain deadline (*DL*). This deadline forces the transmitting node to wait for acknowledgement beyond the loop closure period in case some packets are lost to announce that a receiving node has failed. To address the issue of how acknowledgements process is handled, consider the case if the packets numbered *1, 2, 3,…N* have been transmitted. The transmitting node waits for an acknowledgement from the receiving nodes until it receives an acknowledgement after which it resets its counter, or until the deadline, in those circumstances it discontinues sending packets and declares the end of network life. A false alarm is said to have occurred if all transmitting nodes cease to transmit packets at the end of the deadline even though some receiving nodes are still operative.

5.3. Performance analysis via simulation

This subsection investigates the dependence of network performance on frequency of acknowledgements from the receiving nodes and on the storage capacity of the nodes. Two aspects of storage capacity are considered. As a transmitting node that is responsible for re-transmission, a copy of a transmitted packet is stored for as long as the set deadline for the reception of an acknowledgement of that packet. Since this section does not deal with re-transmission, only a counter is needed. As a receiving node, a received packet may be allowed to wait in a buffer for its turn to be processed at a node while a previously received packet is being processed. Performance consideration includes time to network failure, packet loss rate, and false alarm rate. The wireless network is simulated both with and without feedback. The importance of selection of appropriate deadline, buffer size, and loop closure period to the network performance is delineated.

Designing and analyzing simulation experiment depends on the type of simulation [27]. The performance measures of interest in this study necessitate terminating simulations. The terminating condition for the wireless network materializes when the network fails, which occurs when there is no longer passage of packets from source to sink. This could be due to the failure of all nodes in a cluster, or due to a false alarm that occurs when an acknowledgment deadline is passed even though there are still surviving nodes in the receiving cluster.

For a given scenario, *n* independent replications of a terminating simulation are run where each replication is terminated as soon as a network failure is declared, and is begun with the same initial condition of an empty and fully operative network. The behavior of the network is studied based on apposite data collected in the course of simulation and the performance measures of interest are estimated using the data. As the number of collected data sample *n* increases, that is as $n \to \infty$, the sample mean of the performance estimates from the multiple independent replications converges almost surely to the true mean of the underlying distribution of the performance measure, based on the Strong Law of Large Numbers [8].

The packet inter-arrival time is exponentially distributed with mean time varied at *0.1, 0.25* and *0.35* sec. The service time of each node is fixed at *T = 0.02* sec. The channels are opted to have an independent failure probability of *0.01*. The failure time distribution of a node is Weibull whose shape parameter α and scale parameter β can be determined statistically if the failure data of *L* concurrent tests are available [47]. The mean of Weibull distribution is given by $(\beta/\alpha)\ \Gamma(1/\alpha)$, where Γ is the complete gamma function. For chosen parameter values of $\beta = 50$, $\alpha = 3$ in the *5*- hop wireless network, the mean failure of the node occurs after serving about 2233 packets.

Time to network failure (*TNF*) is defined as the expected number of packets received at the sink before the network fails. Using simulation data, time to network failure is estimated and is characterized as the average of total number of packets received at the sink before the terminating condition occurs over multiple replications. Suppose N_i^R is the total number of packets that reach the sink when the simulation terminates for the i^{th} single run of the network. The simulation is run *n* times with the same initial conditions. Data $N_1^R, N_2^R, \cdots, N_n^R$ are collected and are used to obtain the time to network failure as a point estimate of the form,

$$\hat{TNF}_n = \frac{1}{n}\sum_{i=1}^{n} N_i^R$$

The $(1-\alpha)$ confidence interval of the estimate is given by $[\hat{TNF}_n - t_{n-1,\alpha/2}\sqrt{S_n^2/n}, \hat{TNF}_n + t_{n-1,\alpha/2}\sqrt{S_n^2/n}]$, where

$$\frac{\hat{TNF}_n - E\{TNF\}}{\sqrt{S_n^2/n}}$$

follows a Student's *t* distribution of *n-1* degrees of freedom, and the sample variance is given by

$$S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(N_i^R - \hat{TNF}_n)^2 .$$

This method is called the method of independent replications [1].

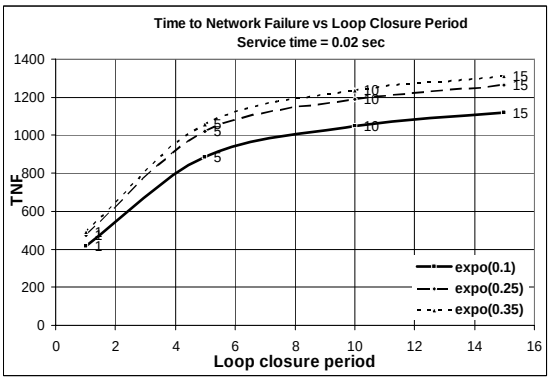
Consider the case when the feedback is applied to the wireless network, with *N=5*, *DL=25*, inter-arrival time an exponential distribution of mean time *0.1 sec*, and the rest of the specifications remain the same. Then, with *n = 30*, the time to network failure is estimated as

$$\hat{TNF}_{30} = \frac{1}{30}\sum_{i=1}^{30} N_i^R = 884.5 \text{ packets},$$

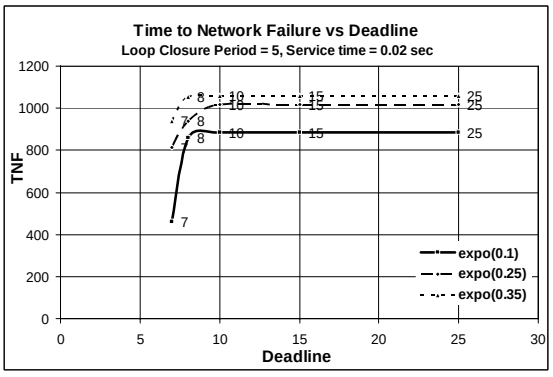
and the half width of 95% confidence interval is determined as

$$t_{n-1,\alpha/2}\sqrt{S_n^2/n} = 105.29 .$$

Thus, the *95%* confidence interval for *TNF* is *779.21≤ TNF ≤ 989.79.*



(a)



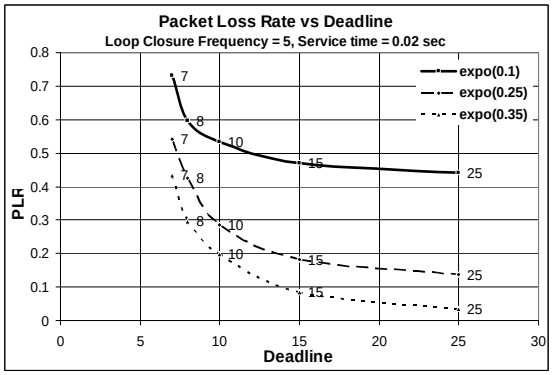
(b)

Figure 5.4 Time to network failure with respect to (a) loop closure period, (b) acknowledgment deadline

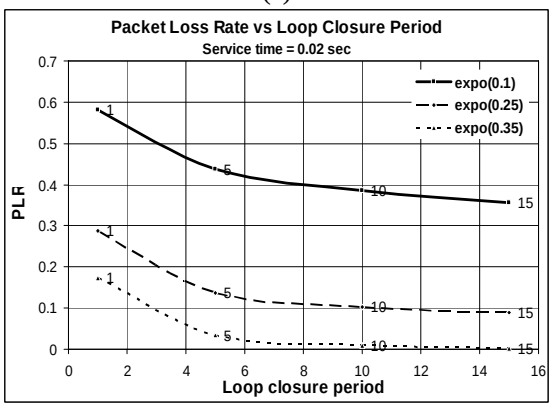
To assure that the selected number of simulation runs is sufficiently large to uphold the central limit theorem, a simple test can be performed to confirm whether the data resembles a normal distribution. If not, more replications are required.

Simulation analysis of the *5*-hop network inclusive of the feedback mechanism indicates, as shown in Figure V.4, that the time to network failure increases with increasing period of loop closure (*N*) for a given deadline, and with increasing deadline for a given loop closure period. The former is attributed to the increased life expenditure associated with more frequent supervisory activities that consume extra power. The latter has to do with reduced false alarm rate as the deadline increases. A transmitting node assumes that a receiving node has failed if it does not receive acknowledgement from the receiving node within a deadline. Hence, false alarm rate increases with a shorter deadline.

In several types of wireless networks, accurate and complete reception of packets is of paramount importance; hence, packet loss rate is a significant performance measure. To estimate the packet loss rate, two sets of data are collected. First set collected from the i^{th} replication is $(N_i^S - N_i^R)$, which is the difference between the total number of packets created in the source and that received in the sink in the i^{h} replication by the time the termination condition is met. The second set collected from the i^{th} replication is N_i^S, the total packets generated in the source by the time the termination condition is met.



(a)



(b)

Figure 5.5 Packet loss rate with respect to (a) loop closure period and (b) deadline

The sample function is defined as

$$\frac{N_i^S - N_i^R}{N_i^S}$$

which represents the percentage loss with respect to total packets generated in the i^{th} replication. The point estimate of packet loss rate can be obtained as

$$\hat{PLR} = \frac{1}{n}\sum_{i=1}^{n}\frac{N_i^S - N_i^R}{N_i^S}$$

From Figure 5.5, it is evident that the packet loss rate decreases with increasing loop closure period, and increasing deadline for all inter arrival rates. This is because a larger loop-closure period implies less node power expenditures in supervisory activities and hence more likely success in transmission, and a extended deadline implies a lowered false alarm rates and hence an effectively longer network life. As inter-arrival time increases, the *PLR* decreases, because the chance of packet collision decreases.

Also of interest is the estimate of false alarm rate (*FAR*), for which the sample function is defined in terms of an indicator function as follows

$$\mathrm{I}_i = \begin{cases} 1, \text{if false alarm occurs in the } i^{\text{th}} \text{ replication} \\ 0, \text{otherwise} \end{cases}$$

Recall that a false alarm occurs when a transmitting node does not receive acknowledgement from any of the receiving nodes at the time of a deadline while some of the receiving nodes are still alive. Then the point estimate for the false alarm rate is defined as

$$\hat{FAR} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{I}_i$$

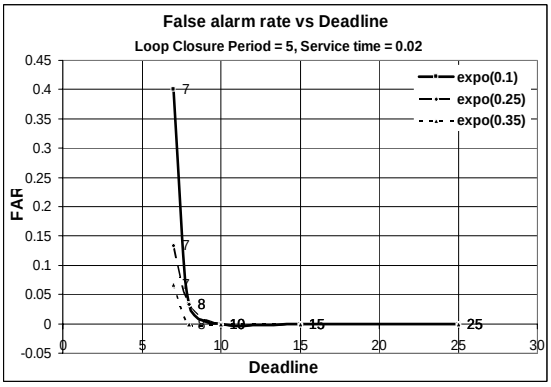


Figure 5.6 False alarm rate with respect to deadline

It is obvious that false alarm rate decreases with increasing deadline until there is no more benefit with further extension of deadline beyond which network fails almost surely before false alarm occurs, as shown in Figure 5.6.

The packet loss rate (*PLR*) is also examined against the buffer size at a node. For the given range of arrival rate, buffer size of *1* is determined to be sufficient, as shown in Figure V.7. The reduction in packet loss rate when a sufficiently large buffer is in place is mainly attributed to the effective avoidance of collision.

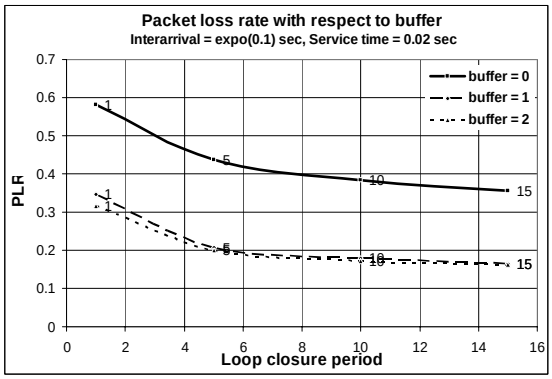


Figure 5.7 Packet loss rate with respect to loop closure period with buffer size as parameter.

5.4. Section summary

In this section, some aspects of performance of a *K* - hop wireless network subject to both channel fading and node failures are evaluated. The goal is to understand the effect of supervisory activities, in particular, the acknowledgement of reception of packets on the number of packets the battery-powered network can transmit and the probability of success of transmission (*1*- packet loss rate).

In the following, the implications of our analysis to network design will be discussed, so will some future work.

The need for acknowledgment arises when there is a benefit for the nodes to know when to stop transmitting. Introducing acknowledgment, however, always incur a cost in terms of both network lifetime and transmission success.

The only way to reduce the impact of the acknowledgement to network lifetime is to keep a sufficiently low loop closure rate that minimizes a combined measure of energy expenditure before and after network failure. The energy expenditure after the network failure is attributed to the additional power consumption in operative nodes due to the delay in applying a stopping rule.

On the other hand, a sufficiently large buffer size can effectively reduce the chance of packet collision, which in turn reduces packet loss rate. In our study, however, the utilization of individual nodes is low. The very high packet loss rate observed is therefore largely attributed to the lack of reliability of the nodes and the channels. The multiple-hop environment accentuates the effect of unreliability. Therefore, higher level of redundancy becomes necessary.

The matter becomes much more complicated if re-transmission of packets is considered. In that case a lower loop closure rate implies a longer delay for a packet to pass the network. Therefore, one must consider trading off delay against extra power consumption and bandwidth contention that come with a more frequent loop closure rate. In this

case, however, as long as the storage capacity is sufficient, packet loss can be reduced to practically none in the early life of the network. Again optimal level of redundancy should be sought with respect to all competing interests to prolong the network life. A protocol for re-transmission is conceivably more complex than the protocol for acknowledgement. Therefore, we propose to consider in the future re-transmission with which network lifetime will be evaluated against energy and bandwidth efficiency, and time delay, and supervisory coverage will be estimated for a given protocol.

6. SUPERVISORY CONTROL OF A DATABASE UNIT

6.1. Problem description

THE recent effort to install and test monitoring tools and to increase the level of redundancy in critical subsystems in air operation centers [45] has provided opportunities for vast performance improvement in its command and control (C2 hereafter) supporting systems. Our previous work on a controlled C2 processing unit [44] has demonstrated that reduced response time to service requests and shortened periods of system unavailability, as a result of automated monitoring and control, can raise significantly the probability to attain the desired outcome in an air operation. This section shifts focus to one other critical C2 subsystem, a database unit. A simulation study [49] has been performed recently using Arena [35], [23] on a controlled database unit. The results indicate, however, that the architecture shown in Figure 6.1 is extremely inefficient, where the service burden rests almost entirely on the primary server, while the secondary server, though indispensable for the required system availability, is rarely utilized.

Figure 6.2 shows an alternative architecture for which the potential improvements in response time and in service availability are to be examined. The partition of the database into multiple sets of data (to be called data classes hereafter), and the simultaneous access to multiple servers allow the reduction of the response time to queries, whereas the presence of a secondary data class in every server leads to fault-tolerance and therefore higher service availability. The performance improvement, however, cannot be achieved in a cost-effective manner without a reconfiguration scheme called a supervisory control that acts on the state information of the database system. This effort investigates several such schemes that differ by their control authorities. To assess the effectiveness of these schemes in a quantified manner, the model in Figure 6.2 (and that in Figure 6.1) is given the interpretation of a queuing network [39] with specific sets of operating policies and structural parameters. The control authorities considered include the ability to restore the lost data and/or the ability to route queries. In order to obtain an analytic model of manageable size for scrutinizing the effects of supervisory control, the archiving process is ignored, and the queuing network is of the closed type [8]. A simulation study is being conducted currently without these simplifications.

The section is organized as follows. Subsection B models the database system in Figure 6.2 as a Markov chain [22] with supervisory control. Subsection C evaluates a set of performance measures under several supervisory control policies. Subsection D concludes the section. Details of the database model are given in Appendix.

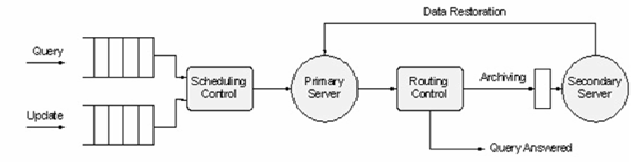


Figure 6.1 Redundant database unit

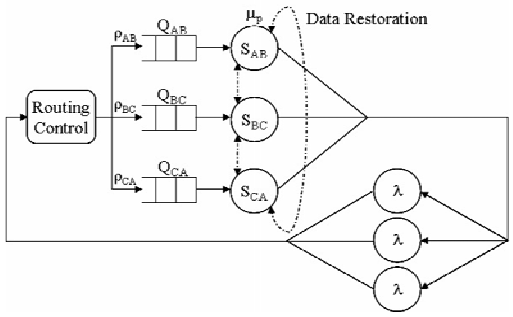


Figure 6.2 Partitioned database unit

6.2. Modeling and control

6.2.1. Modeling

The database unit in Figure 6.2 contains three servers in parallel to answer three classes (A, B, C) of queries for which relevant information can be found in the partitioned sets A, B, C of the database, respectively. Server S_{AB} contains database class *A* as the primary class and database class *B* as the secondary class. Server S_{BC} contains database class *B* as the primary class and database class *C* as the secondary class. Server S_{CA} contains database class *C* as the primary class and database class *A* as the secondary class. The failure of a server implies the loss of two classes of data within the server. A system level failure is declared when two servers fail, in which case one class of data is said to be lost. The queues preceding servers S_{AB}, S_{BC}, and S_{CA} are named Q_{AC}, Q_{BC}, and Q_{CA}, respectively. All queues are of sufficient capacity. Service is provided on a FCFS basis at each server.

The three delay elements imply that there are always three customers present in the unit at any given time. A new query is generated at a delay element upon the completion of the service to a query at one of the servers. The delay elements are intended to be also reflective of the response time to the querying customers by other service nodes in the C2 supporting system, which are not explicitly modeled. Any new query is assumed to be equally likely to seek database class A or B or C. Therefore routing probabilities ρ_{AB}, ρ_{BC}, and ρ_{CA} are assigned the same values under the normal operation condition.

The use of a queuing network model for the database is based on its suitability to involve control actions and our intention to capture their effects on the system performance. The model is built in this study with the premise that event life distributions have been established for the process of query generation $(\exp(\lambda) \equiv 1 - e^{-\lambda t})$, the process of service

completion $(\exp(\mu))$, the process of server failure $(\exp(\nu))$, the process of data restoration $(\exp(\gamma))$, and the process of unit overhaul $(\exp(\omega))$ when the failed database unit is repaired. All such processes are independent. Standard statistical methods that involve data collection, parameter estimation, and goodness of fit tests [53] exist for identifying event life distributions. Since all event lives are assumed to be exponentially distributed, the database unit can be conveniently modeled as a Markov chain specified by a state space $\mathcal{X}$, an initial state probability mass function (pmf) $\pi_x(0)$, and a set of state transition rates Λ, [8] [22]. The reader uninterested in the details of model building can advance to the paragraph right above Equation (15).

6.2.1.1. State space $\mathcal{X}$

A state name is coded with a *6*-digit number indicative of all queue lengths and server states in the unit. With some abuse of notations, a valid state representation is given by $x=Q_{AB}Q_{BC}Q_{CA}S_{AB}S_{BC}S_{CA}$, where queue length Q_{AB}, Q_{BC}, $Q_{CA} \in \{0, 1, 2, 3\}$ with total length $L \equiv Q_{AB}+Q_{BC}+Q_{CA} \le 3$, and server state $S_{AB}, S_{BC}, S_{CA} \in \{0, 1, 2\}$. Server state "*2*" ≡ data are lost in both the primary and the secondary classes in a server, "*1*" ≡ the data in the primary class have been restored and data in the secondary class have not been restored, and "*0*" ≡ data in both primary class and secondary class in a server are intact. A server is said to be in the down state if it is either at state "*1*" or at state "*2*". For example, state *110020* indicates that server S_{AB} is up with one customer in its queue, server S_{BC} is down with both classes of data gone and one customer in its queue, and server S_{CA} is up and idle. Note that the queue length includes the customer being served. There are *540* valid states in the system. The total number of states is reduced to *147* when the states of system level failures are aggregated. The symmetry of the system permits the arrangement of customers in the queues at the time of system level failure to be captured in one of seven states, allowing the system to return to an equivalent state upon completion of the system overhaul. A set of alternative state names are assigned from $\mathcal{X} = \{1, 2, \ldots, 147\}$ with *000000* mapped to $x = 1$ and the aggregated system failure states mapped to $x \in \{141, 142, 143, 144, 145, 146, 147\}$.

6.2.1.2. Initial state pmf $\{\pi_x(0), x = 1,2,\ldots,147\}$

It is assumed that the database unit starts operation from state $x = 1$, i.e., the initial state probability is given by vector $\pi(0) = [1\ 0\ \ldots\ 0]$. When overhaul is considered at the occurrence of a system level failure, the system returns to a state with an equivalent arrangement of customers in the queues once the database unit is renewed [22] and ready for operation again.

6.2.1.3. Set of state transition rates Λ

A transition rate table containing all transition rates is created following a similar procedure as that described in [42], however with a more compact representation. The state transition table is given in Appendix. The list of current states occupies the first column of the table. In the row corresponding to each state, the set of all feasible next states are listed with each next state followed by the rate at which the next state is reached. Events that trigger the transitions and the corresponding transition rates are given as follows. A newly generated query enters one of the servers with rate $\rho^{u_2}(3-L)\times\lambda$ where ρ^{u_2} is a controlled routing probability by control variable u_2. A query is answered at a server with rate μ. A complete data loss occurs at a server with rate ν. Data in the primary data class of a server are restored with rate $\gamma_p u_1$ where u_1 authorizes whether to restore the lost data. Data in the secondary data class of a server are restored with rate $\gamma_s u_1$. Finally, the failed database unit is renewed with rate ωu_3, where u_3 decides whether to repair the failed system. All rates are relative, for their net effects depend on the time unit specified.

Let $X \in \mathcal{X}$ denote the random state variable at time t. The set of state transition functions

$$p_{i,j}(t) \equiv P[X(t) = j \mid X(0) = i], i, j = 1,2,\cdots,147 \qquad (15)$$

for the continuous-time Markov chain can be solved from the forward Chapman-Kolmogorov equation [8]

$$\dot{P}(t) = P(t)Q,\ P(0) = I,\ P(t) = [p_{i,j}(t)]\ , \qquad (16)$$

where Q is called an infinitesimal generator or a rate transition matrix whose $(i,j)^{\text{th}}$ entry is given by the rate associated with the transition from current state i to next state j in the rate transition table. State probability mass function at time t

$$\pi(t) = [\pi_1(t)\quad \pi_2(t)\quad \cdots\quad \pi_{147}(t)],\ t \ge 0 \qquad (17)$$

is computed by

$$\pi(t) = \pi(0)P(t)\,. \qquad (18)$$

At this point a Markov model for the database unit of Figure 6.2 has been established. The state probabilities are the basis for evaluating the performance of the database unit, which is conducted in Subsection C.

6.2.2. Control policies

Our ultimate goal is to eliminate all single point failures, and to mitigate the effects of a single server failure on the performance of the database unit. Our approach is to base the supervisory control actions on the state information, which effectively alter the transition rates when loss of data occurs in a single server.

Taking into consideration the symmetry of the model, the control policy is described only for the case of a failed server S_{AB}. When routing control is effective, the routing probabilities are determined by the state of S_{AB} and by whether the lost data can be restored. Thus, $\rho^{u_2} = \rho_{AB}(S_{AB},u_1),\ \ \rho_{BC}(S_{AB},u_1),\ \ \rho_{CA}(S_{AB},u_1)$ with $\rho_{AB} + \rho_{BC} + \rho_{CA} = 1$. The control policies considered for

this study are summarized as follows.

$$u_1 = \begin{cases} 0, & S_{AB}=2, S_{BC} \text{ serves}, S_{CA} \text{ serves (no restoration)} \\ 1, & \begin{cases} S_{AB}=2, S_{BC} \text{ serves}, S_{CA} \text{ restores class A data} \\ S_{AB}=1, S_{CA} \text{ serves}, S_{BC} \text{ restores class B data} \end{cases} \end{cases}, \quad (19)$$

$$u_2 = \begin{cases} 0, & S_{AB}=2, \rho_{AB}=\rho_{BC}=\rho_{CA}=\frac{1}{3} \\ 1, & \begin{cases} S_{AB}=2, \rho_{AB}(2,u_1), \rho_{BC}(2,u_1), \rho_{CA}(2,u_1) \\ S_{AB}=1, \rho_{AB}(1,u_1), \rho_{BC}(1,u_1), \rho_{CA}(1,u_1) \end{cases} \end{cases}. \quad (20)$$

Four sets of routing probabilities are shown in the following table as examples, where $S_{BC}=0$ and $S_{CA}=0$ are assumed.

Table 6.1 Examples of routing probabilities

u_1	u_2	S_{AB}	ρ_{AB}	ρ_{BC}	ρ_{CA}
0	1	2	0	1/2	1/2
1	0	2 (1)	1/3(1/3)	1/3(1/3)	1/3(1/3)
1	1	2 (1)	0 (1/6)	2/3(1/6)	1/3(2/3)
1	1	2 (1)	0 (0)	1 (0)	0 (1)

The composition of u_1 and u_2 gives rise to four different control policies. The case of $(u_1, u_2) = (0, 0)$ corresponds to the case of a single point failure, and is therefore not considered in the performance analysis. The control policies in the other three cases are named

Policy 1: $(u_1, u_2) = (0, 1)$ when a server is down,
Policy 2: $(u_1, u_2) = (1, 0)$ when a server is down, (21)
Policy 3: $(u_1, u_2) = (1, 1)$ when a server is down.

Note that policy 2 does not permit routing, whereas policy 1 does not permit restoring. As can be seen, policy 3 allows variations in the routing probabilities to the intact servers. A special consideration with the case $u_1=0$ is the rerouting of the customers who have arrived at a server before the server fails to the delay elements.

The presence of supervisory control in the transition rate table is seen via u_1, u_2, u_3, $n_1 = 1-u_1$, $n_2 = 1-u_2$, and $n_3 = 1-u_3$. The values of u_1, u_2, u_3 represent specific control actions associated with data restoration, query routing, and unit overhaul, respectively.

6.3. Performance analysis

6.3.1. Time to system failure

When $u_3 = 0$, the Markov chain model for the database unit contains seven absorbing states $x \in \{141, 142, 143, 144, 145, 146, 147\}$ at which the chain remains forever once it is entered. These are the states of system level failure. The rest of the 140 states are transient states. Decompose the state probability vector

$$\pi(t) \equiv [\underbrace{\pi_\tau(t)}_{1\times 140} \quad \underbrace{\pi_\alpha(t)}_{1\times 7}], \quad (22)$$

where vector $\pi_\tau(t)$ contains the transient state probabilities, and $\pi_\alpha(t)$ are the absorbing state probabilities. Decomposing the rate transition matrix Q and the state transition function matrix $P(t)$ solved from (2) accordingly yields

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ 0 & 0 \end{bmatrix}, P(t) = \begin{bmatrix} P_{11}(t) & P_{12}(t) \\ 0 & I \end{bmatrix}. \quad (23)$$

From (2), (4), and (9), it can be determined that the probability density function of time to system failure, or time to absorption, is given by

$$\dot{\pi}_\alpha(t) = \pi_\tau(0) P_{11}(t) Q_{12}, \pi_\alpha(0) = 0, \quad (24)$$

where

$$\pi_\tau(0) = [1 \quad 0 \quad \cdots], \; P_{11}(t) = e^{Q_{11}t}. \quad (25)$$

In addition, the mean time to failure of the database unit can be shown to be [22].

$$MTTF = -\pi_\tau(0) Q_{11}^{-1} 1_\tau, 1_\tau = [1 \quad \cdots \quad 1]^T \quad (26)$$

Figure 6.3 below shows the dependence of mean time to failure of the database unit on the restoration rate.

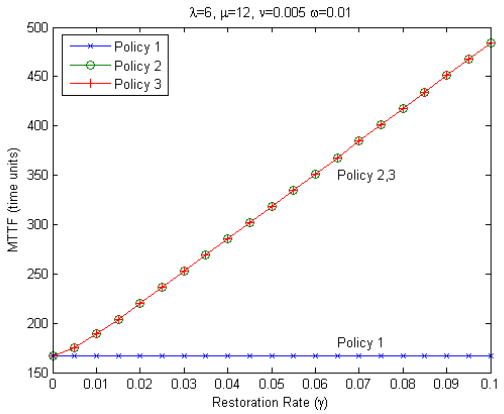


Figure 6.3 Database unit mean time to failure versus restoration rate

6.3.2. Steady-state availability

Suppose as soon as the database unit reaches a system level failure, an overhaul process starts. Suppose with a rate ω the unit is repaired, and at the completion of the repair, the unit immediately starts to operate again. In this case u_3 is set to 1 in the model, whereas it is set to 0 in the case of an absorbing chain. The existence of a unique steady-state distribution of the Markov chain when $u_3=1$ is guaranteed if the chain is irreducible (or ergodic) [22]. Ergodicity is satisfied under policy 2 and policy 3. Although ergodicity is not met under policy 1 without eliminating the few unreachable states in this case, a unique steady state distribution is obtained nevertheless in our computation. The steady state availability, which can be roughly thought of as the fraction of time the database unit is up, is given by

$$A_{sys} = 1 - \pi_F(\infty), \quad (27)$$

where $\pi_F(\infty)$ is the sum of the system level failure state probabilities, determined by solving

$$\pi(\infty) Q = 0, \text{ and } \sum_{x=1}^{147} \pi_x(\infty) = 1. \quad (28)$$

Figure 6.4 shows the steady-state availability as a function of restoration rate at a fixed overhaul rate. Figure 6.6 demonstrates the benefit of the success in supervisory control to steady-state availability.

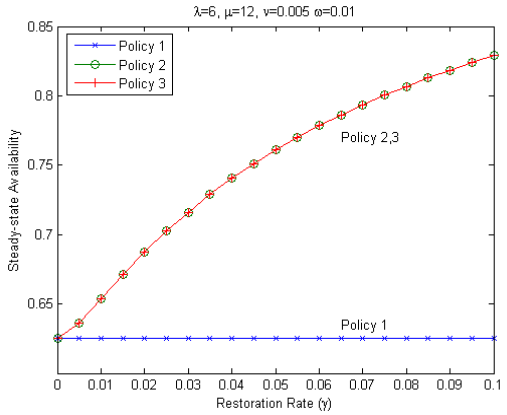


Figure 6.4 Steady-state availability of the database unit versus restoration rate

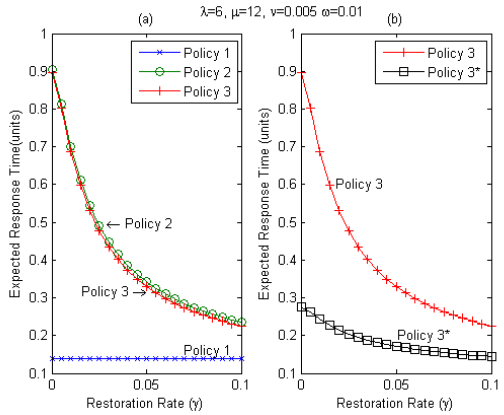


Figure 6.5 Average query response time versus restoration rate

6.3.3. *Response time*

The average response time $E[R]$ is the expectation of the ratio of total amount of time that all customers spend in the upper portion of the system to the number of customers that are serviced. A loose argument is given below to justify the way $E[R]$ is computed in this subsection. Define the vector C where $c(i)$ is the number of customers in the system at state i. The numerator of $E[R]$ is then $\pi(\infty)Ct$. Computing the number of customers that are serviced requires counting the number of transitions from one state to another that have occurred that have introduced a new customer to the system. Define a matrix N such that $n(i,j)$ is equal to the number of customers introduced into the system when the system transitions from state i to state j. The total number of transitions for a given i and j is then

$$T(i,j) = tN(i,j)\pi_i(\infty)Q(i,j). \tag{29}$$

Therefore, the average response time $E[R]$ of the system is taken as

$$\frac{\pi(\infty)Ct}{\sum_{i=1}^{147}\sum_{j=1}^{147}T(i,j)} = \frac{\pi(\infty)C}{\sum_{i=1}^{147}\sum_{j=1}^{147}N(i,j)\pi_i(\infty)Q(i,j)}. \tag{30}$$

Figure 6.5a and Figure 6.6 show the average response time as a function of restoration rate with the overhaul rate fixed, and a function of overhaul rate with the restoration rate fixed, respectively, for all three policies. The routing probabilities in rows 1 through 3 in Table 6.1 are in fact used for calculating all performance measures resulting from Policies 1 through 3, respectively. Policy 1 enjoys a lower response time because the intact servers need not deny customers in order to restore the failed server. Also, customers present at the time of server failure in policy 1 are emptied into the delay elements and incur no response time gains.

Figure 6.5b shows the effect of applying Policy 3*: routing all customers to the intact server that is not restoring the failed server, an alternative to Policy 3. The reduced response time in policy 3* results from customers not waiting at a failed server. This policy may not be as advantageous in a system of higher traffic intensity.

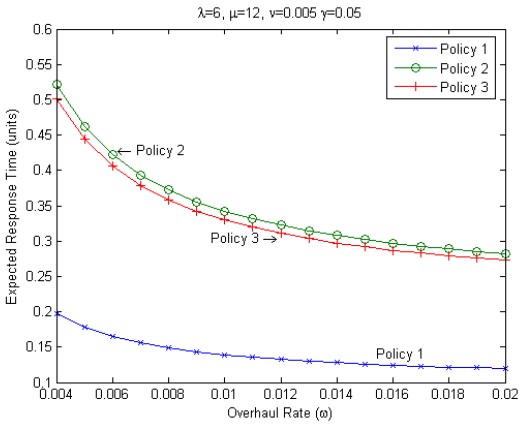


Figure 6.6 Average query response time versus overhaul rate

6.3.4. *Overhead*

Overhead is a quantity introduced to reflect the ratio of the time invested on helping the database unit to survive longer to its overall busy time. It is a measure of the cost of supervisory control. More specifically,

$$\theta \equiv \frac{\Pr[S_{AB} \text{ restores or fails} \mid \text{unit is not failed}]}{\Pr[S_{AB} \text{ restores or fails or serves} \mid \text{unit is not failed}]} \tag{31}$$

Overhead θ is calculated for both the absorbing chain ($u_3 = 0$) as a function of time, and the irreducible chain ($u_3 = 1$) as a function of server failure rate. These are shown in Figure 6.7 and Figure 6.8. In Figure 6.7, it is seen that restoration incurs a higher overhead in the early life of the unit. As the database unit ages, its server becomes more likely to fail. A control policy that permits restoration becomes advantageous. There is a reduction in overhead across all polices with an increase in the arrival rate because of the resulting increased utilization. In Figure 6.8, for sufficiently low server failure rate, overhead is always lower with restoration. When server failure rate passes some threshold, however, restoration becomes expensive. Overhead is

expected to gain more significance as a function of time and a function of server failure rate when the server life distributions have an increasing failure rate, such as in the case of Weibull distribution.

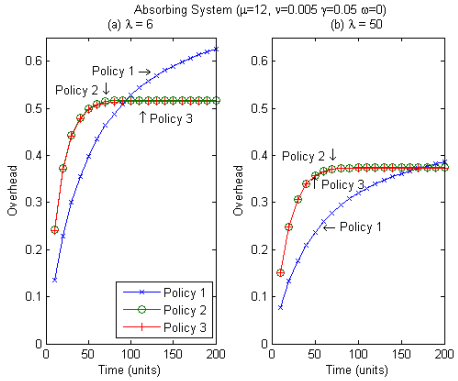


Figure 6.7 Overhead versus time in the absorbing chain

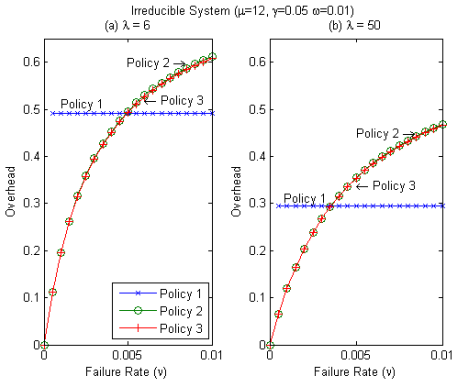


Figure 6.8 Overhead versus failure rate in the irreducible chain

6.4. Section summary

This section modeled a redundant database unit in C2 for investigation of fault-tolerance and responsiveness afforded by a set of supervisory control policies. In all the performance measures examined, restoration (u_1) is more effective than routing (u_2). It is expected that when the number of queries increase, or the traffic becomes more intensive, the effectiveness of routing will be more apparent.

The study presented in this section is limited by our ability to deal with complex problems analytically. Most restrictive is the size of the state space. The closed-queuing network model shown in Figure 6.2 presents perhaps the smallest possible state space for which the investigation on control policies is nontrivial. Besides answering queries, the database unit also must be updated from time to time. In that case, two types of service requests exist and the state space must be expanded. Almost equally restrictive is the assumption that times to event occurrence are exponentially distributed. Since there is only one parameter in an exponential distribution, it is likely to be unsuitable to truthfully describe some of the processes. Discrete event simulations are being carried out where the simplifying assumptions are removed to substantiate our claims on the benefit of supervisory control under more general settings in terms of the types of services, the number of customers, and the types of distributions of event lives.

Also ongoing is the extension of this study to incorporate the effect of decision and control under uncertainty and time delay due to, for example, incomplete state information and the time required for state estimation, respectively. The results will be reported in a future section.

6.5. Acknowledgment

Both authors thank Ms. Sudha Thavamani, a student at Binghamton University pursuing her Ph.D. degree under N. Eva Wu, for her assistance in the simulations with Arena [49] that supported the preliminary study [44] of the database unit shown in Figure 6.1. N. Eva Wu also thanks Dr. Timothy Busch at the AFRL Rome Research Site for his insights in many sessions of discussion on the issue of architecture of the command and control supporting systems.

7. A Simulation Study of the Effect of Supervisory Control on a Redundant Database Unit

7.1. Problem description

THE focus of this work is on studying the effect of supervisory control [8] on a number of important measures that pertain to C2 system performance with a redundant architecture first proposed and studied in [51], where a database is partitioned in a way that allows multiple servers to process customers in parallel with information backed-up throughout the system. The proposed architecture is shown in Figure 7.1, where the data are partitioned into the sets A, B, and C. Customers entering the system are routed based on the type of information they require.

To enhance fault-tolerance in the face of crash and site failure, and improve the responsiveness to queries, supervisory control is applied to the partitioned database unit. The response time and availability can be potentially improved by strategically routing customers based on the state of the servers. Supervisory control introduces policies that allow the restoration of lost data and/or the routing of queries based on the state of the information in the system.

The objectives of this work are to qualitatively analyze the performance of the partitioned database unit under supervisory control and varying structural parameters, in terms of MTTF, availability, response time, and overhead, based on the results obtained through discrete event system (DES) simulation.

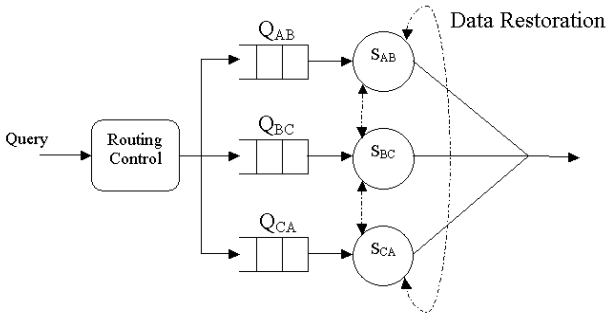


Figure 7.1. Partitioned database unit

7.2. Background

The presentation in this section is drawn from [51] to recapitulate aspects of modeling, control, and performance analysis of the database unit shown in Figure 7.2 [51], to identify the limitations of the analytical method employed there, and to briefly describe the extensions made in this section.

7.2.1. System Model

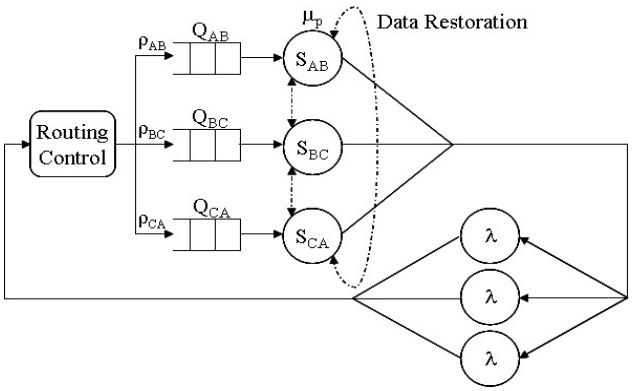


Figure 7.2. Closed queuing network model

The database unit to be studied is taken from [51], which is intended to be representative of a C2 supporting system. A closed queuing network representation of the unit is shown in Figure 7.2. The information contained within the system is partitioned into sets A, B, and C and placed on three servers that exist in parallel to answer three classes of queries A, B, and C, respectively. Server S_{AB} contains class A primary data and class B secondary data. Server S_{BC} contains class B primary data and class C secondary data. Server S_{CA} contains class C primary data and class A secondary data. When a server fails, both its primary and secondary data are lost. A server is "down" when either class of data are lost, and a system failure occurs when two servers are down concurrently.

The queues preceding S_{AB}, S_{BC}, and S_{CA} are named Q_{AB}, Q_{BC}, and Q_{CA}, respectively. They are of sufficient size that no queries are lost or blocked and operate on a first come, first serve (FCFS) basis.

The delay elements, each labeled λ indicating an average delay 1/λ, are representative of the response times incurred at other nodes of the C2 supporting system which are not modeled here. The three elements imply that there are only three customers in the system at any given time, a limitation of the Markov model in [51] that is to be removed in this study. Upon completion of processing at a server, a customer returns to one of its delay elements, and after a period of time, re-enters the system. Each time a customer enters the system it is equally likely to require information of class A, B, or C. Therefore, under normal operating conditions, the routing probabilities ρ_{AB}, ρ_{BC}, and ρ_{CA}, where $\rho_{AB} + \rho_{BC} + \rho_{CA} = 1$, are given the same values.

The model is built with the premise that event lifetime distributions have been established for all the processes involved. The delay process, or equivalently, query generation, has an exponential distribution $\exp(\lambda) \equiv 1\text{-}e^{-\lambda t}$, where λ is the rate and 1/ λ is the mean. The same is true for the process of service completion ($\exp(\mu_p)$), the process of server failure ($\exp(\nu)$), the process of data restoration ($\exp(\gamma)$), and the process of unit overhaul ($\exp(\omega)$), when the entire unit is repaired due to system failure. All processes are

independent. Note that all rates and therefore means are relative and carry the units time^{-1} and time, respectively.

7.2.2. Control Policies

To maximize the efficiency of the database unit under server failures, two supervisory control inputs are introduced based on the state information of the system. These control actions alter the transition rates of the system when data loss occurs in a server for the purpose of improving performance. The necessary state information is the current state of the servers. Define server state S_{AB}, S_{BC}, $S_{CA} \in \{0,1,2\}$ where "2" ≡ both the primary data and the secondary data are lost in a server, "1" ≡ the primary data have been restored but the secondary data have not yet been restored, and "0" ≡ the primary data and secondary data in the server are both intact. A server is failed, or in the down state, when either class of data are lost, and up, when both the primary and secondary data are intact.

Two supervisory control inputs, u_1 and u_2, govern restoration and routing, respectively. The control input u_1 allows an intact server to halt its current process and restore lost data in a failed server, and input u_2 adjusts the routing probability of customers based on the state of the servers. Because of the symmetry of the model, the control inputs and policies may be sufficiently described by the case of only one failed server S_{AB}, where the remaining two servers must be intact for the system to be up. The control inputs may be summarized as follows.

$$u_1 = \begin{cases} 0, & S_{AB} = 2, S_{BC} \text{ serves}, S_{CA} \text{ serves (no restoration)} \\ 1, & \begin{cases} S_{AB} = 2, S_{BC} \text{ serves}, S_{CA} \text{ restores class A data} \\ S_{AB} = 1, S_{CA} \text{ serves}, S_{BC} \text{ restores class B data} \end{cases} \end{cases}, \quad (32)$$

$$u_2 = \begin{cases} 0, & S_{AB} = 2, \rho_{AB} = \rho_{BC} = \rho_{CA} = \frac{1}{3} \\ 1, & \begin{cases} S_{AB} = 2, \rho_{AB}(2,u_1), \rho_{BC}(2,u_1), \rho_{CA}(2,u_1) \\ S_{AB} = 1, \rho_{AB}(1,u_1), \rho_{BC}(1,u_1), \rho_{CA}(1,u_1) \end{cases} \end{cases}. \quad (33)$$

Recall the routing probabilities ρ_{AB}, ρ_{BC}, and ρ_{CA}. Under supervisory control, these probabilities are dependent not only on the routing control input u_2 and the state of the servers, but also on the restoration control input u_1. Table 7.1 shows three sets of routing probabilities.

Table 7.1 Examples of routing probabilities

u_1	u_2	S_{AB}	ρ_{AB}	ρ_{BC}	ρ_{CA}
0	*1*	*2*	*0*	*1/2*	*1/2*
1	*0*	*2 (1)*	*1/3(1/3)*	*1/3(1/3)*	*1/3(1/3)*
1	*1*	*2 (1)*	*0 (1/6)*	*2/3(1/6)*	*1/3(2/3)*

The composition of u_1 and u_2 gives rise to four different control policies. The case of $(u_1, u_2) = (0, 0)$ corresponds to the case of a single point failure, and is therefore not considered in the performance analysis. The control policies in the other three cases are named

Policy 1: $(u_1, u_2) = (0, 1)$ when a server is down,
Policy 2: $(u_1, u_2) = (1, 0)$ when a server is down, (34)
Policy 3: $(u_1, u_2) = (1, 1)$ when a server is down.

Note that policy 2 does not permit routing, whereas policy 1 does not permit restoring. A special consideration with the case $u_1=0$ is the rerouting of the customers who have arrived at a server before the server fails to the delay elements.

7.2.3. Analytic Results

The above system was first studied in **Error! Reference source not found.** where the database unit was modeled as a closed Markovian queuing network. The performance of the system under supervisory control was evaluated based on the performance measures of mean time to failure (MTTF) of the system, steady-state availability, expected response time, and service overhead.

System failure is defined as the loss of a second server before the restoration of a first failed server is completed. MTTF is a measure of the life of the system. The MTTF was found to significantly improve under policies 2 and 3, apparently attributed to the introduction of restoration. Availability, a measure of the percentage of time the system is available to serve customers (i.e., not in a system failure state), also improved under these policies.

The expected response time, defined as the length of time a query spends in the upper portion of the system shown in Figure 7.2, also benefited from policies 2 and 3 for a sufficiently high restoration rate γ. However, at low values of γ, the system profited from not having to devote a majority of its resources to restoring failed servers but simply servicing customers with its two intact nodes under policy 1. Policy 3 showed slightly better performance than policy 2; this advantage is expected to improve with the addition of customers to the system.

Overhead is defined as the cost incurred by the system for self-preservation. It is calculated as the ratio of time invested in restoring the system to its overall busy time, and does not include the overhaul process. As the failure rate ν increased, policies 2 and 3 became expensive, and surpassed the overhead associated with policy 1.

7.2.4. Limitations

The Markov model of the database unit presented in [51] suffered many limitations. The complexity of the model was restricted by the need for a manageable number of states and the exponential event lifetime distributions. A linear increase in either the number of customers or the number of servers allowed in the system causes an exponential growth in the number of states, , whereas any non-exponential event lifetime distribution destroys the memoryless properties essential to a Markov model, although many of the processes under consideration are most adequately described by non-exponential distributions.

A majority of database units require a periodic update to the information contained within the system to maintain the currency of the data. As seen in Figure 7.2, [51] omitted the updating process to avoid the explosion of the size of the state space due to the additional class of customers and the non-Poisson nature of the update requests. .

Modeling the database unit by means of a simulation tool enables us to remove the limit on the number of customers, diversify the event lifetime distributions, and include the

update process.

7.3. Simulation Model

7.3.1. Discrete Event System Simulation

Modeling of systems in which the state variable changes only at a discrete set of points in time is known as discrete event system (DES) simulation [1]. Simulation implies solving for the system variables through numerical rather than analytic methods. Observations of the variables collected throughout the history of the model are stored and processed to evaluate system performance measures. A major component in a discrete event system simulation is the future event list which contains the notices for all future events scheduled to occur. For each event that occurs, beginning with the first event of the simulation, durations are either computed or drawn from a statistical distribution, and the end-event is added to the future event list. The advantage of this method is that every time instance need not be evaluated, allowing the simulation to omit time intervals where the state of the system does not change.

The simulation package used in this study is Arena® Professional Edition [35]. Arena® utilizes an object-based design for graphical model development [1]. Objects called modules are used to model system logic and physical components such as servers and queues. In addition, Arena® provides methods for statistical distributions, failure modeling, statistics collection, and process analysis. Arena® allows any number of independent replications to be run for a simulation, with the replication terminating upon a user defined condition. System as well as user defined statistics are collected for each replication and evaluated for the entire simulation.

In this study, the effect of supervisory control is evaluated for MTTF, system availability, expected response time, and overhead as in [51]. MTTF is evaluated using the method of independent replications, whereas system availability, expected response time, and overhead are evaluated using the method of regenerative simulations [27]. All calculated performance measures are obtained from simulation with 100 replications unless otherwise noted. The Process Analyzer is a tool in Arena® that allows a series of simulations (scenarios) with varying system parameters (controls) to be run automatically in succession and displays the chosen system outputs (responses). This feature proved extremely useful in the evaluation of multiple performance measures as a function of varying system parameters. As in [51], supervisory control is evaluated based upon MTTF, system availability, expected response time, and overhead.

7.3.2. Model Verification

The model shown in Figure 7.2 and described in Section II-A was simulated via Arena® under the supervisory control policies presented in Section II-B. The results from each modeling method were compared for verification purposes. The MTTF and availability corresponded between the two simulation methods, as did the system overhead. However, the expected response time calculated from simulation was significantly lower due to the fact that the simulation calculation is not a steady-state measure. Response time statistics are only able to be collected for customers that enter and exit the system. Customers that are trapped outside a failed system do not contribute to the calculated response time. Therefore, only customers who are in a server when the system fails will suffer the delay of the overhaul process. The limited number of customers makes this delay insignificant, resulting in lower response times for the simulation model. This deficiency no longer exists when an open queuing system is introduced in the next section.

With a simulation model constructed and verified in Arena®, the limitations on the number of customers, the event lifetime distributions, and the update process discussed previously, may now be removed, as presented in the following section.

7.4. Analysis via Simulation

7.4.1. Open queuing network

A fixed number of customers severely limits our ability to fully observe the behavior of the system, but it is necessary to model the database analytically. Simulation modeling removes this restriction, and more realistic measures of system performance are provided.

The system is modeled in Arena® as the open queuing network shown in Figure 7.1 where customers enter the system with an exponentially distributed inter-arrival time exp(λ). The customers are removed from the system upon service completion.

The MTTF and availability of the open queuing network are statistically indistinguishable from that obtained in the closed simulation model. The expected response time however does increase significantly now that customers are freely allowed to enter and accumulate in the system. The benefits of routing control are apparent, as shown in Figure 7.3. Policy 3 realizes a lower response time with customers routed strategically by supervisory control input u_2. However, as failed servers are restored at a higher rate, the advantage of policy 3 decreases. Routing control becomes less beneficial because queue lengths do not grow as large at failed servers.

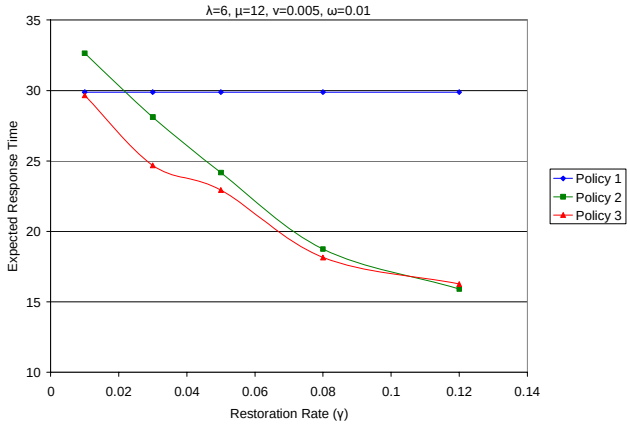


Figure 7.3. Expected response time of open queuing network versus restoration rate

Overhead is less sensitive to the type of queuing network. The values shown in Figure 7.4 correspond to those obtained analytically in the closed queuing network. The overhead of policy 1 is unaffected by an increase in the rate of failure because the system is never required to restore itself. Restoration is beneficial at low failure rates, however, beyond some threshold, it becomes expensive to the system.

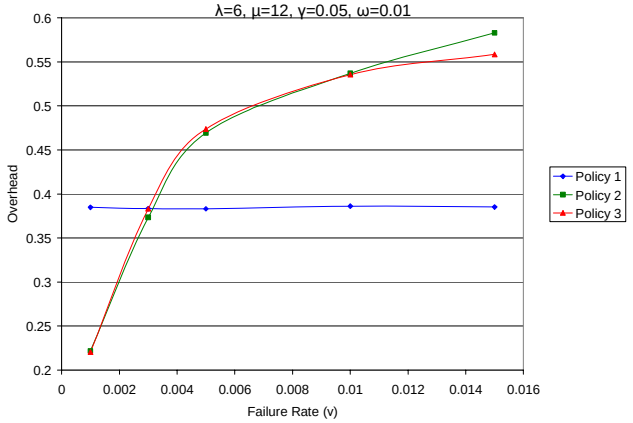


Figure 7.4. Overhead of open queuing network versus failure rate

7.4.2. Generalized Distributions

Simulation of the database unit permits the removal of the limitation to exponentially distributed event lifetimes. The exponential distribution has a constant failure rate and therefore is unfit for many event lifetimes. For example, a component with a failure process described by an exponential distribution has a constant failure rate and is therefore probabilistically always as good as new, regardless of its age [39], while in reality, most components are more likely to fail as they age. For the distributions described below, parameters such as the shape parameter α and the scaling parameter β, are chosen to provide a mean equivalent to that of the exponential distribution previously used.

The arrival process lifetime remains exponentially distributed (exp(λ)). Often systems undergo certain "busy" periods, but for the purposes of this study, customers will arrive at a constant rate. The gamma distribution is often used to represent the time required to complete a task [23] and is therefore used to describe the process of service completion (gamma(1/αβ=μ)). A component lifetime is better described by a distribution that reflects the age of the component. The Weibull distribution has a rate that varies with time, and is used to describe the failure process (Weibull(α,1/β=ν)). For α > 1, the probability of failure increases with age. Triangular distributions with mode *m* are used for the restoration process (tria(1/*m*=γ)) and the overhaul process (tria(1/*m*=ω)) because these event lifetimes are relatively deterministic. The time required to restore a known amount of data should not vary significantly.

With the failure event lifetime now dependent on time, the MTTF and availability of the system, as shown in Table 7.2, is expected to decrease. This is true for the policies involving restoration; however it is not the case for policy 1, which is unaffected. For policies allowing restoration, the MTTF is dependent on a failed server recovering before another failure occurs. A time dependent failure rate causes failures to occur more closely (assuming the lifetimes begin concurrently), increasing the likelihood of overlapping failures, and reducing the MTTF. Policy 1 is unaffected because the failure of a second server always results in a system failure. The MTTF obtained for policy 1 under each type of distribution is statistically equal because the mean values of both distributions are equal.

Table 7.2 Comparison of exponential and generalized distributions

	MTTF		Availability	
Policy	Exponential	Generalized	Exponential	Generalized
1	*169*	*172*	*.61*	*.62*
2	*317*	*244*	*.71*	*.69*
3	*317*	*265*	*.71*	*.69*

Expected response time decreases under the generalized event lifetime distributions, as shown in Figure 7.5, however the values follow the same trend as those shown in Figure 7.3.

Overhead is shown in Figure 7.6 for the generalized distributions. Comparison with Figure 7.4 shows a decrease over that observed from the use of exponential distributions. Using the Weibull distribution, servers are more likely to fail as they age, resulting in less time devoted to restoration in the early stages of their lifetime. As the system ages, more simultaneous failures are likely to occur. When failures occur close together, the time a server spends restoring another server decreases because the overhaul process takes over to restore the system.

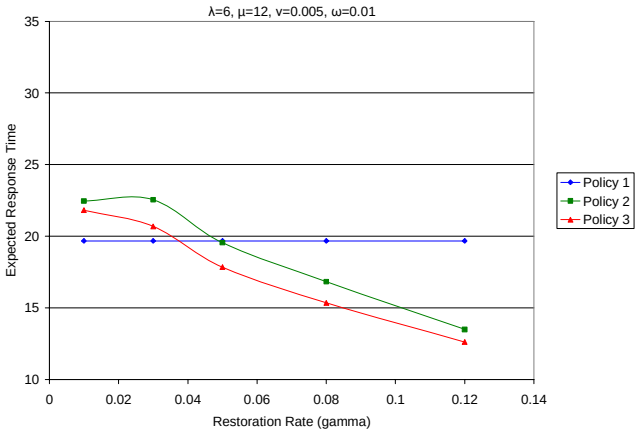


Figure 7.5. Expected response time versus restoration rate using generalized distributions

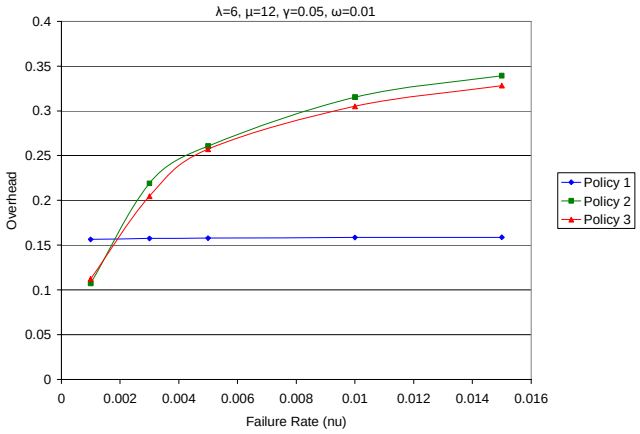


Figure 7.6. Overhead versus failure rate using generalized distributions

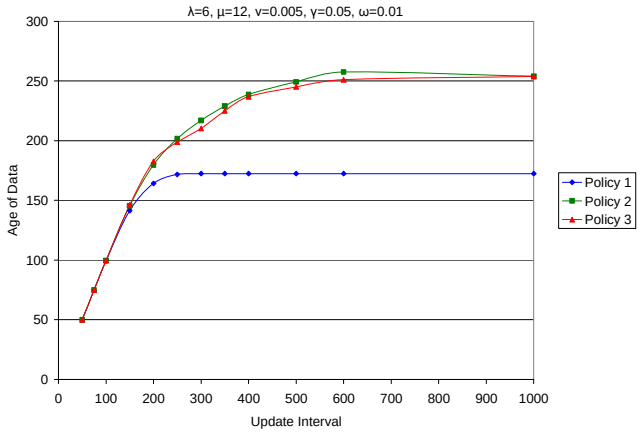


Figure 7.7. Age of data versus the update interval

7.4.3. System Update Process

In order to keep the information stored in the database unit current and useful, the system must be periodically updated. While an overhaul of the system will update the data, system failure should occur infrequently, resulting in the need for a system update at a steady interval in the form of an update entity.

Update entities arrive at a deterministic rate, with the inter-arrival time being the acceptable age of the information in the system. An update entity is sent to each server and becomes the first in-line at the queue. Both the primary data and secondary data for each server are contained on the update entity. It follows that the time to process an update entity is described by twice the restoration process distribution. A server is unavailable while processing an update entity. An update entity arriving at a failed server will restore that server, a significant benefit to policy 1, as well as the remaining policies, under which servers no longer have to restore a failed server that is processing an update entity. It is important to note that an update to a server does not reset the lifetime of the component.

The update interval only partially determines the age of the data in the unit. The data, on average, will only be as old as the minimum of the update inter-arrival time or the MTTF, which will result in the system being overhauled with current data. The age of the data contained within the database unit is shown in Figure 7.7 versus the update interval. At low intervals, the data is as old as the inter-arrival time. As the interval increases, the age of the data reaches a maximum of the MTTF of each policy given in Table 7.2. At these high intervals, the data is being updated only by the system overhaul process.

The update interval has a significant impact on the availability of the system, as shown in Figure 7.8. As the update interval increases, the system is more available to answer queries. Availability increases until the update interval is so large it neither improves MTTF nor hinders processing queries, and it reaches the steady-state value given in Table 7.2. Policy 1 enjoys a higher availability at low update intervals because failed servers are being restored by the update process, which, on average, is 2.5 times faster than the overhaul process. Beyond an update interval equal to its MTTF, policy 1 no longer benefits from restoration provided by the update process and its availability diminishes slightly.

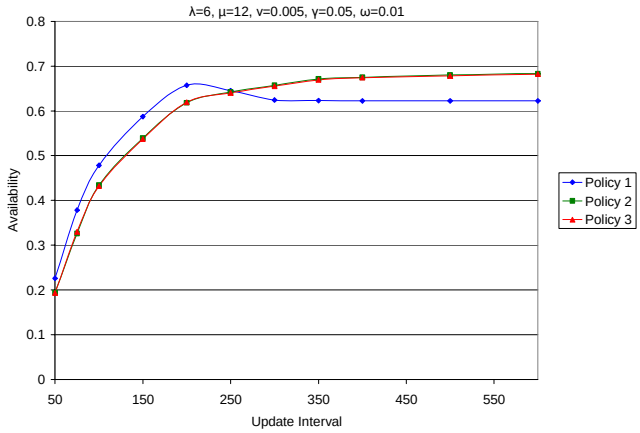


Figure 7.8. Availability versus the update interval

Frequent system updates tax the resources of the database unit causing an increase in the expected response time, as shown in Figure 7.9. As the update interval increases, the customer service interruption caused becomes negligible, and the expected response time reaches the values shown in Figure 7.5. Policy 1 experiences a substantial increase in expected response time at a low update interval because many times only two servers are available to process queries. The impact on expected response time is more severe when those servers are interrupted to process update entities.

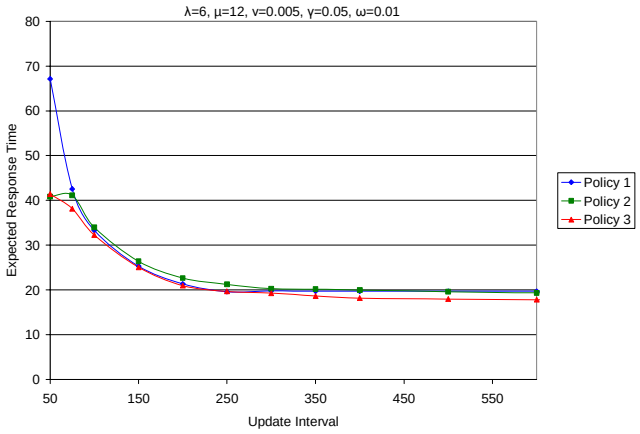


Figure 7.9. Expected response time versus the update interval

Updating the system is considered time invested in maintaining the database unit. Therefore, the expression for overhead θ in **Error! Reference source not found.** is modified to

$$\theta \equiv \frac{\Pr\left[S_{AB} \text{ restores or fails or updates} \middle| \text{unit is not failed}\right]}{\Pr\left[S_{AB} \text{ restores or fails or updates or serves} \middle| \text{unit is not failed}\right]} \quad .(4)$$

Overhead improves as the update inter-arrival time increases, as shown in Figure 7.10. Restoration of failed servers by an update entity is not assessed as overhead for policy 1 because the database unit is failed during this time. Therefore, overhead is significantly lower for policy 1.

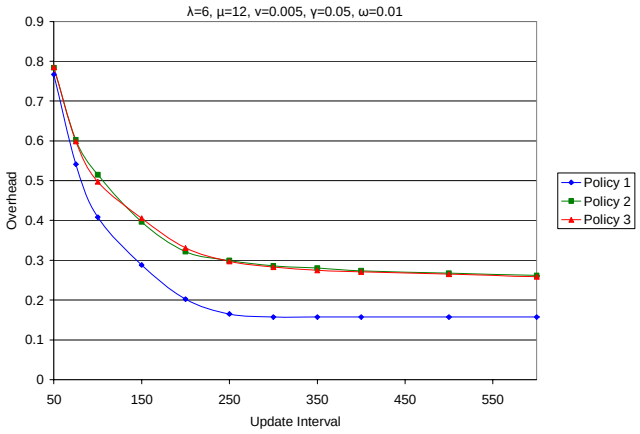


Figure 7.10. Overhead versus the update interval

7.5. Section summary

The use of discrete event system simulation allows the removal of limitations imposed by having to represent a database unit analytically as a Markov model. A more practical system may be evaluated that includes an open queuing network, dynamic event lifetime distribution rates, and an update process. This section has modeled and evaluated a database unit representative of a C2 supporting system under several supervisory control policies. The effects of restoration (u_1) and routing (u_2) were assessed based on measures of fault-tolerance and responsiveness. While restoration remains more beneficial than routing, the benefits of routing control are slightly more visible for an unlimited-sized population as compared to the results obtained in [51]. The addition of an update process, while necessary, weighs heavily on the performance of the system.

7.6. Acknowledgment

The first two authors thank Ms. Sudha Thavamani, a student at Binghamton University pursuing her Ph.D. degree under N. Eva Wu, for her assistance in the simulations with Arena®.

8. Performance of a Controlled Database Unit Subject to Decision Errors and Control Delays

8.1. Problem description

A recent effort to install and test monitoring tools and to increase the level of redundancy in critical subsystems in air operation centers has provided opportunities for vast performance improvement in its command and control supporting systems. Our previous work on a controlled processing unit [44] has demonstrated that reduced response time to service requests and shortened periods of system unavailability, as a result of automated monitoring and control, can raise significantly the probability to attain the desired outcome in an air operation. A more recent study by Wu, Metzler, and Linderman [51] on a database unit as shown in Figure 8.1 further revealed the benefits of a conscientious design of redundant architecture, and the application of supervisory control, which were measured in terms of the mean time to unit failure, the steady state availability, the expected response time, and the service overhead of the database unit.

To assess the performance in a quantified manner, both the processing unit [44] and the database unit (Figure 8.1) [51] were given the interpretation of a queuing network **Error! Reference source not found.**, [24] with specific sets of operating policies and structural parameters. The control authorities considered included the ability to restore the first failed server, and the ability to route service requests. In order to obtain an analytic model of manageable size for scrutinizing the effects of supervisory control, the queuing network was restricted to the closed type **Error! Reference source not found.**, [24]. In addition, all the event lifetime distributions were assumed to be exponential. A simulation study was conducted by James Metzler et al., [32] using Arena [23], [53] with all the above restrictions removed.

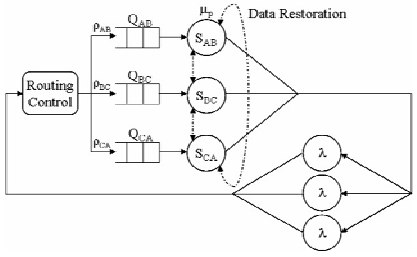


Figure 8.1 A partitioned database unit

An underlying assumption of the existing study is that the state information in the queuing network model of a given unit is known exactly at any given time. In reality, however, it is not practical to monitor every state variable. As a result, the knowledge on a certain set of states is inferred based on the observables. On the other hand, control actions are likely required at the time of a state transition, such as the occurrence of a component failure, in which case a process of diagnosis must take place before a state-based control action. The time required for diagnosis can be random, and the outcome of the diagnosis can be uncertain. The objectives of this section, therefore, are to seek for ways to incorporate the effects due to decision errors and control action delays into the Markov model of a queuing network, and to use the model to access the impact of such errors and delays on the performance of the database unit in Figure 8.1.

The section is organized as follows. Subsection B describes the baseline model of the controlled database unit in Figure 8.1. Subsection C discusses our approaches to modeling the effects of control delays and decision errors. Subsection D presents the results of performance evaluation parameterized with respect to the amount of control action delay and the probability of error.

8.2. Baseline model for a controlled database unit

The description of the baseline model, i.e., the model that does not include decision errors and control delays, follows to a large extent that of Wu, Metzler and Linderman [51]. The database unit in Figure 8.1 contains three servers in parallel to answer three classes (A, B, C) of queries for which relevant information can be found in the partitioned sets A, B, C of the database, respectively. Server S_{AB} contains database class *A* as the primary class and database class *B* as the secondary class. Server S_{BC} contains database class *B* as the primary class and database class *C* as the secondary class. Server S_{CA} contains database class *C* as the primary class and database class *A* as the secondary class. The failure of a server implies the loss of two classes of data within the server. A system level failure is declared when two servers fail, in which case one class of data is completely lost. The queues preceding servers S_{AB}, S_{BC}, and S_{CA} are named Q_{AC}, Q_{BC}, and Q_{CA}, respectively. All queues are of sufficient capacity. Service is provided on a FCFS basis at each server.

The three delay elements of average delay 1/λ imply that there are always three customers present in the unit at any given time. A new query is generated at a delay element upon the completion of the service to a query at one of the servers. The delay elements are intended to be also reflective of the response time to the querying customers by other service nodes in the system that are not explicitly modeled. Any new query is assumed to be equally likely to seek database class A or B or C. Therefore routing probabilities ρ_{AB}, ρ_{BC}, and ρ_{CA} are assigned the same values.

The use of a queuing network model for the database is based on its suitability to involve control actions and to capture their effects on the system performance. The model is built in this study with the premise that event life distributions have been established for the process of query generation $(\exp(\lambda) \equiv 1 - e^{-\lambda t})$, the process of service completion $(\exp(\mu))$, the process of server failure $(\exp(\nu))$, the process of data restoration $(\exp(\gamma))$, and the process of unit overhaul $(\exp(\omega))$ when the failed database

unit is repaired. All such processes are independent. Standard statistical methods that involve data collection, parameter estimation, and goodness of fit tests exist for identifying event life distributions. Since all event lives are assumed to be exponentially distributed, the database unit can be conveniently modeled as a Markov chain specified by a state space X, an initial state probability mass function (pmf) $\pi_x(0)$, and a set of state transition rates Λ [8], [22]. The reader uninterested in the details of model building can advance to the paragraph right above Equation (1).

8.2.1. State space X

A state name is coded with a *6*-digit number indicative of all queue lengths and server states in the unit. With some abuse of notations, a valid state representation is given by $x=Q_{AB}Q_{BC}Q_{CA}S_{AB}S_{BC}S_{CA}$, where queue length Q_{AB}, Q_{BC}, $Q_{CA} \in \{0, 1, 2, 3\}$ with total length $L \equiv Q_{AB}+Q_{BC}+Q_{CA} \le 3$, and server state $S_{AB}, S_{BC}, S_{CA} \in \{0, 1, 2\}$. Server state "*2*" ≡ data are lost in both the primary and the secondary classes in a server, "*1*" ≡ the data in the primary class have been restored and data in the secondary class have not been restored, and "*0*" ≡ data in both primary class and secondary class in a server are intact. A server is said to be in the down state if it is either at state "*1*" or at state "*2*". For example, state *110020* indicates that server S_{AB} is up with one customer in its queue, server S_{BC} is down with both classes of data gone and one customer in its queue, and server S_{CA} is up and idle. Note that the queue length includes the customer being served. There are *540* valid states in the baseline system. The total number of states is reduced to *141* when all the states of system level failures are aggregated. A set of alternative state names are assigned from $X = \{1, 2, \ldots, 141\}$ with *000000* mapped to *x=1* and the aggregated system failure state mapped to *x=141*.

~~2)~~8.2.2. *Initial state pmf* $\{\pi_x(0), x=1,2,\ldots,141\}$

It is assumed that the database unit starts operation from state *x=1*, i.e., the initial state probability is given by vector $\pi(0) = [1\ 0\ \ldots\ 0]$. When overhaul is considered at the occurrence of a system level failure, all customers are flushed out to the delay elements. Once the database unit is renewed and ready for operation again, it starts at the same initial state *x=1*, and a renewal process [22] is formed.

~~3)~~8.2.3. *Set of state transition functions* $p_{i,j}(t)$

Events that trigger the transitions and the corresponding transition rates are given as follows. A newly generated query enters one of the servers with rate $(3-L)\times\lambda/3$. A query is answered at a server with rate μ. A complete data loss occurs at a server with rate ν. Data in the primary data class of a server are restored with rate $\gamma_p u_1$, and data in the secondary data class of a server are restored with rate γ_s, where u_1 authorizes whether to restore the lost data for the primary class. Finally, the failed database unit is renewed with rate ωu_3 where u_3 decides whether to repair the failed system.

Let $X \in \mathcal{X}$ denote the random state variable at time *t*. The set of state transition functions

$$p_{i,j}(t) \equiv P[X(t) = j \mid X(0) = i],\ i, j = 1,2,\cdots,141 \qquad (35)$$

for the continuous-time Markov chain can be solved from the forward Chapman-Kolmogorov equation [23]

$$\dot{P}(t) = P(t)Q(u_1,u_3),\ P(0) = I,\ P(t) = [p_{i,j}(t)]\ , \qquad (36)$$

where $Q(u_1,u_3)$ is called an infinitesimal generator or a rate transition matrix whose $(i,j)^{th}$ entry is given by the rate associated with the transition from current state *i* to next state *j* in the rate transition table. State probability mass function at time *t*

$$\pi(t) = [\pi_1(t)\quad \pi_2(t)\quad \cdots\quad \pi_{141}(t)],\ t \ge 0 \qquad (37)$$

is computed by

$$\pi(t) = \pi(0)P(t)\,. \qquad (38)$$

At this point a baseline Markov model for the database unit of Figure 8.1 has been established. Since transition rate matrix *Q* is dependent on control actions, the state transition functions $p_{i,j}(t)$ are being controlled, and so are the state probabilities.

8.2.4. Restoration and overhaul

Our ultimate goal is to eliminate all single point failures, and to mitigate the effects of a single server failure on the performance of the database unit. Our approach is to base the supervisory control actions on the state information, which effectively alter the transition rates when loss of data occurs in a single server.

Taking into consideration the symmetry of the model, the control policy is described only for the case of a failed server S_{AB}. The control policies considered for this study are summarized as follows.

$$u_1 = \begin{cases} 0, & S_{AB} = 2, S_{BC} \text{ serves}, S_{CA} \text{ serves (no restoration)} \\ 1, & S_{AB} = 2, S_{BC} \text{ serves}, S_{CA} \text{ restores class A data} \end{cases} \qquad (39)$$

The presence of supervisory control in the transition rate matrix is seen via u_1, u_3, $1\text{-}u_1$, and $1\text{-}u_3$. The values of u_1, u_3 represent specific control actions associated with data restoration, and unit overhaul, respectively. Unit overhaul occurs only at the unit failure state *141*.

The complete baseline model is provided in [51] in the form of a rate transition table, where an additional control variable u_2 was present. u_2 controls routing probabilities when data loss occurs in a server. u_2 is removed in this section because the small number of queries in the system makes the additional benefit afforded by routing control less obvious to observe.

8.3. Model augmentation to include errors & delays

This subsection focuses on modeling the effects of decision errors and control action delays upon entering a state. These two undesirable effects can be intertwined. To quantify their individual impact on performance, they are separated into the class of decision errors when a control action is taken incorrectly but immediately upon entering a state, and the class of delayed control actions when a correct control action is taken but after some time delay. In addition, there are deterministically diagnosable systems for which the only cost of diagnosis is time [8]. Two augmented models

will be generated in this subsection representing a controlled database unit with decision error, and one with control action delay, respectively. Each model will contain *201* states.

8.3.1. Effect of decision error

The supervisory control considered in this study is state information-based. Upon entering a state, say, *A*, any information deficiency can result in uncertainty in decision making as to whether to take a control action or what control actions to take. In this case, every decision carries a risk.

An example of a decision error with the database unit would be that upon a server failure a wrong server is being identified as having failed. More specifically, S_{AB}, for instance, has failed. S_{CA}, however, is mistakenly thought to be the failed one. Based on the false information, the control action would be for S_{BC} to restore data class *C* in S_{CA}, whereas S_{AB} would be expected to continue to work. As a consequence of a wrong decision, none of the servers can process queries for a period of time. The database unit is said to have entered an intermittent error state. It is assumed that from this state, only transitions to more server failures, or to the recovery to original destination state can occur. Figure 8.2 depicts a generalized representation of such a case.

Without loss of generality, let *A* be a state that is entered upon a total data loss in a server. Let *C* be the state entered upon the completion of primary database restoration associated with the data loss. Let B_1 through B_n be the states representing completions of services at other *n* servers. Let G_1, …, G_l be the state entered upon the arrival of a new query in one of the server queues. Let F_1 through F_m be the states entered upon data loss at other *m* servers. The notion of intermittent state *I* is introduced, as shown in Figure 8.2, to allow the representation of imperfect decision making upon entering *A.* Therefore, there is an intermittent error state for each state that involves outgoing transitions with weakened control authorities due to some decision errors. In the database unit of Figure 8.1, altogether *60* states are added to the original *141* state baseline model. Note that states G_i's are not shown explicitly in Figure 8.2, and they can be regarded as part of F_i's from this point on. It is assumed that once the primary database restoration takes place for a particular server, the secondary restoration is error free.

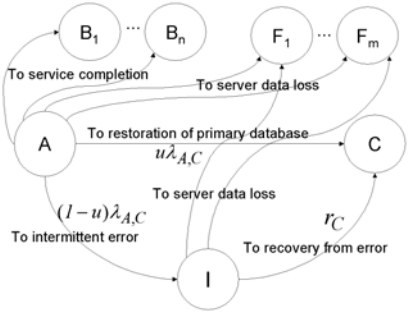


Figure 8.2 Decision error modeling w. an intermittent error state

Let $\lambda_{A,C}$ denote the transition rate from state *A* to state *C* in the absence of decision error to restoration of primary database associated with the most recent data loss. Let *u* be the probability of successful restoration given that the event of restoration occurs. $(1-u)$ then is referred to as the *thinning* [8] of the Poisson arrival process associated with the restoration. The split of rate $\lambda_{A,C}$ into rate $u\lambda_{A,C}$ and rate $(1-u)\lambda_{A,C}$ is sometimes also called a decomposition [22] of a Poisson arrival process into type 1 with probability *u* and type 2 with probability (*1-u*).

An imperfect decision corresponds to the value of *u* being less than unity. As a consequence, the authority of supervisory control that is supposed to reinforce the restoration process has been weakened. The smaller the value of *u*, the weaker the control authority is.

The rate of recovery from decision error is denoted by r_C. To state the fact that recovery from an intermittent error state to restoration cannot be faster than the error-free (*u=1*) restoration process, $r_C \le \lambda_{A,C}$ is enforced. On the other hand, the outgoing transition rates from the intermittent error state to the states of data loss in other servers, i.e., from *I* to F_i, *i=1, 2, … , m,* are bounded below by the corresponding rates going from *A* to F_i. These transitions further reduce the likelihood of reaching state *C*.

It is now shown that decision errors always degrade the performance in terms of the state transition probability P_{AC} which is the probability that restoration to state *C* occurs given that the state is *A*. It turns out that this probability is readily obtained for a Markov chain [8].

$$P_{AC} = \frac{u\lambda_{AC}}{\Lambda(A)} \ , \tag{40}$$

where

$$\Lambda(A) = \lambda_{AB_1} + \cdots + \lambda_{AB_n} + \lambda_{AF_1} + \cdots + \lambda_{AF_m} + \lambda_{AC} \tag{41}$$

without decision error, in which case $u = 1$ in (6), and

$$\Lambda(A) = \lambda_{AB_1} + \cdots + \lambda_{AB_n} + \lambda_{AF_1} + \cdots + \lambda_{AF_m} + u\lambda_{AC} + (1-u)\lambda_{AC} \tag{42}$$

with decision error, in which case $u < 1$. The denominators of (41) and (42) are the same. Apparently, (40) is proportional to *u*, and is the largest at *u=1* when there is no decision error. On the other hand, flow balance at state *I* yields

$$\dot{\pi}_I = (1-u)\lambda_{A,C}\pi_A - (\sum_{i=1}^{m} \lambda_{I,F_i} + r_C)\pi_I \ , \tag{43}$$

from which the following expression for $\pi_I(t)$ in terms of $\pi_A(t)$ at steady state is obtained

$$\pi_I(\infty) = \frac{(1-u)\lambda_{AC}}{\sum_{i=1}^{m}\lambda_{IF_i} + r_C}\pi_A(\infty) \ . \tag{44}$$

(44) is proportional to *1-u*.

Some results of numerical calculation will be presented in Subsection D based on the state-augmented model of the

database unit of Figure 8.1 that show how certain performance measures depend on the probability of the restoration decision error.

8.3.2. Effect of delayed control actions

Time required for diagnosis can be regarded as the universal cause of a control action delay. Time delay can be traded off in some applications with the decision error to minimize their combined effects. This subsection focuses on the discussion of the effect of time delay alone.

An example on the control action delay with the database unit of Figure 8.1 would be that a total loss of data on a server is not immediately observed. As a result, the action of data restoration is delayed.

As in the previous subsection, let *A* be a state that is entered upon a total loss of data in a server. Let *C* be the state entered upon the completion of primary database restoration associated with the data loss. States B_1 through B_n, and states F_1 through F_m also follow the earlier definitions. Figure 8.3 depicts a proposed model capable of describing a delayed restoration action by an exponentially distributed random amount with average δ^{-1} upon entering state *A*.

In a more general case, there can be an *N*-phased delay implemented in the augmented model by inserting *N* states D_1 through D_N in series between states *A* and *C*. Each state D_i retains outgoing transitions to all B_1 through B_n, and F_1 through F_m, in addition to transition to D_{i+1}. The total amount of delay before restoration action is bounded below by random variable $D = D_1 + \cdots + D_N$, with a generalized Erlang distribution [22]

$$L^{-1}\{\sum_{i=1}^{N} \frac{\delta_i}{s+\delta_i}\}. \tag{45}$$

One may use an *N*-stage Erlang to approach a constant delay, or an *N*-stage hyper-exponential to approach a highly uncertain delay, or a mixture of the two to acquire more general properties [8].

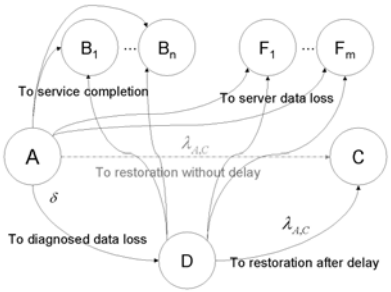


Figure 8.3 Control delay modeling w. a single-stage delay state

Note that there are two significant differences between the decision error model of Figure 8.2 and the control delay model of Figure 8.3. First, the link *to restoration of primary database* is present in Figure 8.2 with a smaller likelihood of transition, whereas the link *to restoration without delay* is absent in Figure 8.3. In addition, all links *to service completion* are absent in Figure 8.2, but present in Figure 8.3. Therefore, these are two cases of different nature.

With a single-stage delay for each state entered upon a total loss of data in a server, *60* states are added to the baseline model. Numerical results on the effect of control action delay will be presented in the next subsection.

8.4. Performance analysis and discussion

8.4.1. Time to system failure

When $u_3=0$, the augmented Markov chain model for the database unit contains one absorbing state $x=201$ at which the chain remains forever once it is entered. This is the state of system level failure. The rest of *200* states are transient states. Decompose the state probability vector

$$\pi(t) \equiv [\underbrace{\pi_\tau(t)}_{1\times 200} \quad \underbrace{\pi_\alpha(t)}_{1\times 1}], \tag{46}$$

where vector $\pi_\tau(t)$ contains the transient state probabilities, and $\pi_\alpha(t)$ is the absorbing state probability. Decomposing the rate transition matrix *Q* and the state transition function matrix *P*(*t*) solved from (36) accordingly yields

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ 0 & 0 \end{bmatrix}, P(t) = \begin{bmatrix} P_{11}(t) & P_{12}(t) \\ 0 & 1 \end{bmatrix}. \tag{47}$$

From (36), (38), and (46), it can be determined that the probability density function of time to system failure, or time to absorption, is given by

$$\dot{\pi}_\alpha(t) = \pi_\tau(0) P_{11}(t) Q_{12}, \pi_\alpha(0) = 0, \tag{48}$$

where

$$\pi_\tau(0) = [1 \quad 0 \quad \cdots], \; P_{11}(t) = e^{Q_{11}t}. \tag{49}$$

In addition, the mean time to failure of the database unit can be shown to be [8]

$$MTTF = -\pi_\tau(0) Q_{11}^{-1} 1_\tau, 1_\tau = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{50}$$

Figure 8.4 below shows the dependence of mean time to failure of the database unit on probability of correct control action for data restoration with restoration rate γ as a parameter. The plot indicates that MTTF is sensitive to restoration rate, and becomes more sensitive to supervisory control coverage at a higher restoration rate. The relative robustness of MTTF with respect to supervisory control coverage can be attributed to the fact that recovery has taken a most optimistic path with $r_C = \lambda_{A,C}$, after a decision error has been made.

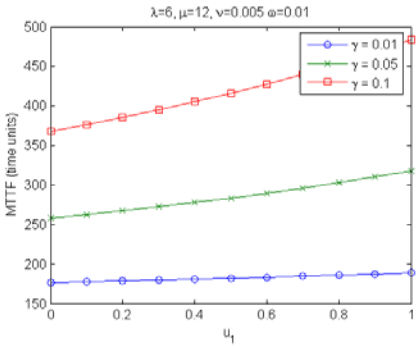


Figure 8.4 Unit MTTF versus control coverage

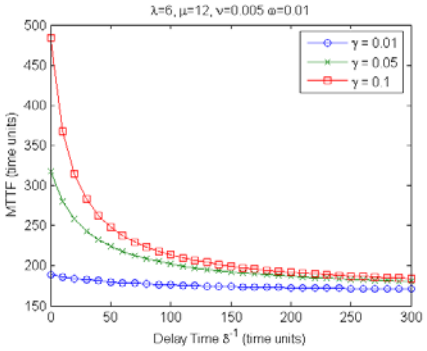


Figure 8.5 Unit MTTF versus control delay

Figure 8.5 above shows the dependence of mean time to failure of the database unit on expected control action delay for data restoration with restoration rate γ as a parameter. It is expected that control action delay affects MTTF more drastically when restoration rate is high. Control action delay becomes dominant in how long it takes to restore data when it becomes comparable to average time required to perform data restoration.

8.4.2. Steady-state availability

Suppose as soon as the database unit reaches a system level failure, an overhaul process starts with all the customers flushed out to the delay elements. Suppose with a rate ω the unit is repaired. At the completion of the repair to condition $\pi(0)$, the unit immediately starts to operate again. In this case u_3 is set to 1 in the model, whereas it is set to 0 in the case of an absorbing chain. The existence of a unique steady-state distribution of the Markov chain when u_3=*1* is guaranteed if the chain is irreducible (or ergodic) [22]. The steady state availability, which can be roughly thought of as the fraction of time the database unit is up, is given by

$$A_{sys} = 1 - \pi_{201}(\infty), \tag{51}$$

where $\pi_{201}(\infty)$ is determined by solving

$$\pi(\infty)Q = 0, \text{ and } \sum_{x=1}^{201}\pi_x(\infty) = 1. \tag{52}$$

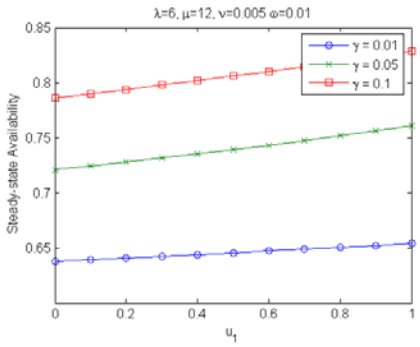


Figure 8.6 Steady-state availability versus control coverage

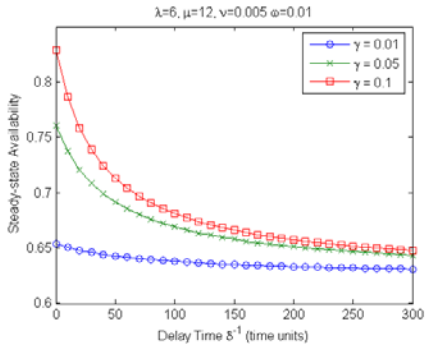


Figure 8.7 Steady-state availability versus control delay

Figure 8.6 and Figure 8.7 show the steady-state availability as a function of supervisory control coverage and a function of expected control action delay. It can be seen that both long delays and slow restoration reduce the availability to unacceptable levels. Explanations on the insensitivity of the availability with respect to coverage and delay under slow restoration conditions follow those for Figure 8.4 and Figure 8.5.

8.4.3. Response time

Consider again the irreducible chain studied in the previous subsection. Let $I_{i,j}$ be the indicator function associated with transition from state *i* to state *j*, and q_{ij} be the corresponding entry in transition rate matrix *Q*. Let N_i be the total number of queries in queue at state *i*. Then the total expected number of queries in queue at steady-state is given by

$$E[X] = \sum_{i=1}^{201}\pi_i(\infty)N_i\ , \tag{53}$$

and the arrival rate at steady-state is

$$\lambda_s = \sum_{i=1}^{201}\pi_i(\infty)\sum_{j=1}^{201}I_{ij}q_{ij}\ . \tag{54}$$

The calculation of the response time at steady-state then follows Little's Theorem $E[X] = \lambda_s E[R]$.

Figure 8.7 and Figure 8.8 show the average response time as a function of supervisory control coverage and a function of control action delay, respectively. Unlike the other performance measures, the sensitivity of the average response time remains relatively significant at a low

restoration rate.

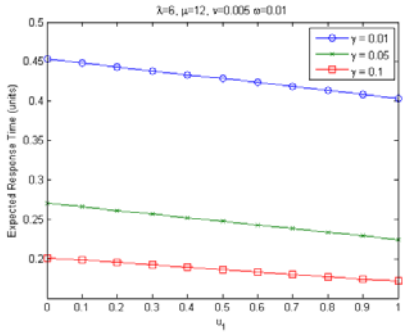


Figure 8.8 Average query response time versus control coverage

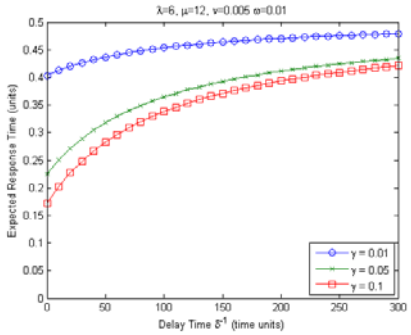


Figure 8.9 Average query response time versus supervisory control delay

8.4.4. Overhead

Overhead is a quantity introduced to reflect the ratio of the time invested on helping the database unit to survive longer to its overall busy time. It is a measure of the cost of supervisory control. More specifically,

$$\theta \equiv \frac{\Pr[S_{AB} \text{ restores or fails} \mid \text{unit is not failed}]}{\Pr[S_{AB} \text{ restores or fails or serves} \mid \text{unit is not failed}]} \qquad (55)$$

Overhead θ is calculated for the irreducible chain (u_3=*1*) as a function of supervisory control coverage and a function of supervisory control delay. These are shown in Figure 8.10 and Figure 8.11. As in the case of availability, overhead at the steady-state becomes unacceptably high at low restoration rate. It is also sensitive to control coverage and delay when restoration rate is high.

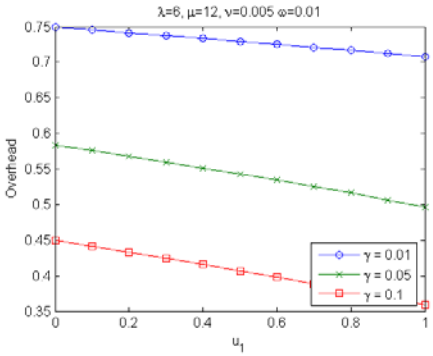


Figure 8.10 Service overhead versus control coverage

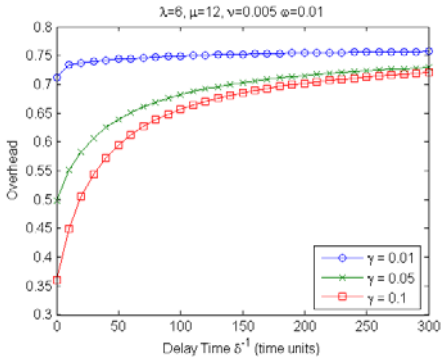


Figure 8.11 Service overhead versus control delay

9. Performance Analysis of a Single Queue Database Unit Subject to Control Delays and Decision Errors

9.1. Problem description

PREVIOUS work by Wu, Metzler, and Linderman [51] on a redundant database unit, as shown in Figure 9.1, demonstrated the advantages of using supervisory control in conjunction with the design of a redundant architecture. Improvements were observed in the mean time to system failure, the steady state availability, the expected response time, and the service overhead of the database unit. On the other hand, much progress has been made in diagnosability of discrete event systems, i.e., whether it is possible to detect a failure occurrence after a finite delay, and in active acquisition, i.e., resource allocation in terms of use of sensors for state estimation [37]. Recognizing that it is impractical to assume perfect knowledge of state information at all times, Wu, Metzler, and Linderman [50] instituted the idea of incorporating decision errors or control delays into modeling the database unit, assuming that probability of a decision error is known and the length of control action delay is random with a known distribution.

The goal of this section is to analyze the same set of performance of a database unit with a configuration shown in Figure 9.2, in the presence of decision errors and control delays. When multiple servers provide service to a single class of customers, it is known that system time generally benefits from committing a customer to a server at the last possible instant [8]. The architecture of in Figure 9.2 reflects our intention to capitalize on such possible benefits in a multiple class and multiple server system, and on potentially more effective use of redundancy. In addition, unlike the earlier study [50] where the effect of decision errors and that of control delays were investigated separately, this section integrates the effects of decision errors and control action delays into a single Markov model.

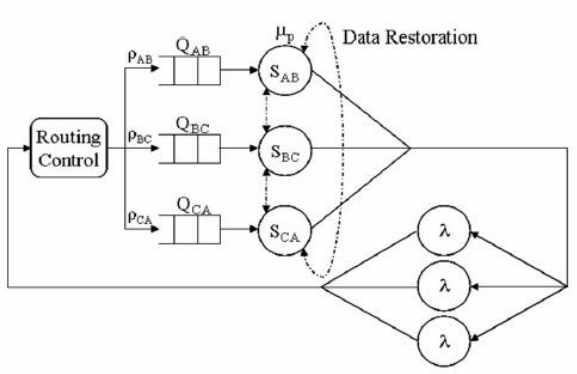


Figure 9.1 Original architecture of a partitioned database unit

.

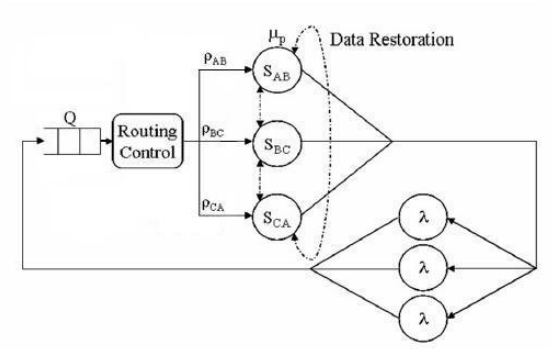


Figure 9.2 Alternative architecture of a partitioned database unit

The organization of this section is similar to that of [50] from which most background material is also drawn. In particular, Subsection B describes the model of the controlled database unit of the new architecture as seen in Figure 9.2, also taking into consideration of control action delays and errors. Subsection C defines the performance measures and presents the numerical results of system performance evaluated with respect to the average delay of control action and the probability of decision error. Subsection D draws concludes.

9.2. Modeling

9.2.1. Baseline model

The database unit presented in Figure 9.2 consists of three servers in parallel. In this configuration, the database server is partitioned into three classes. Each class is given a designation of A, B, or C. Each of the three servers is meant to serve one specific data class (primary class) as well as backup another data class (secondary class). Thus, server S_{AB} has class A as its primary class, while class B is its secondary class. Server S_{BC} has class B as its primary class, while class C is its secondary class. Server S_{CA} has class C as its primary class, while class A is its secondary class. A server failure implies the loss if both the primary and secondary data classes. If server S_{AB} fails, server S_{CA}'s secondary class (A) can be used to restore server S_{AB}'s primary class (A). Once this is completed, server S_{BC}'s primary class (B) restores server S_{AB}'s secondary class (B). There are instances where a secondary is allowed to serve queries, which will be outlined shortly. A failed server always has its primary data class restored before its secondary data class. The entire database unit fails when two server failures occur, resulting the loss of an entire class of data. All servers are fed by a common queue. This queue is of sufficient capacity to contain all queries. The service hierarchy for queries located in the queue depends upon how control variables are defined.

Once a query leaves a server it vanishes into a delay element, where after an average time delay of $1/\lambda$ units a new query of an independent class emerges at the end of the queue. The delays are meant to represent the response time of other elements in the system outside of the database that are not explicitly modeled [45]. The use of three delay elements indicates that there will be no more than three

queries in the unit at any given moment. Once leaving a delay element, a query has an equal probability of requesting class A, B, or C data.

This system is viewed as a queuing network, where event life distributions are dependent on the process of query generation $\left(\exp(\lambda) \equiv 1-e^{-\lambda t}\right)$, service completion $(\exp(\mu))$, server failure $(\exp(\nu))$, data class restoration $(\exp(\gamma))$, and system overhaul $(\exp(\omega))$ which occurs when the unit fails. All of these processes are independent. Since all event lives are assumed to be exponentially distributed, a Markov chain is resulted. This chain has a state space χ, initial state probability mass function (pmf) $\pi_x(0)$, and a set of state transition rates Λ [8].

(a) State Space χ

A state name is created as an alphanumeric string of 6 characters to indicate the status of the queue as well as the status of the servers in the system. States are represented as $x=Q_3Q_2Q_1S_{AB}S_{BC}S_{CA}$, where $Q_i \in \{0, A, B, C\}$ represents whether the i^{th} queue location, $i = 1,2,3$, is unoccupied, or occupied with a query of class *A*, *B*, or *C*; and $S_{\alpha\beta} \in \{I, W, R, P, S, F\}$, $\alpha\beta$=*AB, BC, CA,* represents the state of a server. In particular, "*I*" ≡ server is idle, "*W*" ≡ server is answering a query, "*R*" ≡ sever is restoring another server, "*P*" ≡ primary class of server lost (which implies a concurrent loss of the secondary class), "*S*" ≡ primary class of server restored but secondary class of server not yet restored. "*F*" ≡ failure of database unit. The unit fails when two servers lose class data concurrently. As an example, state *0AAWII* indicates that there are 3 queries in the system: a query is receiving service in server S_{AB}, a query of class *A* waiting in queue position 1, and a query of class *A* is waiting in queue position 2. When examining the system using the $x=Q_3Q_2Q_1S_{AB}S_{BC}S_{CA}$ convention, there are 195 valid states. These states can then be mapped to the set $\chi = \{1,2,3,...,195\}$, where x=*000III* is mapped to $\chi_{0=1}$, etc.

(b) Initial state pmf $\{\pi_x(0), x = 1,2,3,...,195\}$

It is assumed that the system always begins in state 1. Therefore the initial state probability is $\pi(0) = [1\ 0\ 0 \ldots 0]$. In the case of a system failure, queries that were being served are considered 'lost' and sent back to the delay elements, conformable to a preemptive policy [24]. However, those elements residing in the queue remain until the system has been restored. New queries can arrive while the system is not functioning. Once the system is restored, it returns to the appropriate state, dictated by the number of queries currently in queue, and what data class those queries require. As an example, *AABFFF* would transition to *00AWWI*.

(c) Set of transition rates $p_{ij}(t)$

A transition rate matrix is created for all states within the system in a manner outlined in **Error! Reference source not found.**. The rates that are included in this matrix are as follows. Queries arrive with a rate of (3-L)λ, where L is the queue length in addition to the number of queries currently being served. Queries are served with rate μ. A server fails, resulting in complete data loss, with rate ν. The primary data class of a server is restored with rate γ_p. The secondary data class of a server is restored with rate γ_s. After the unit completely fails, it is repaired with rate ω.

Let $X \in \chi$ be the random state variable at time *t*. The set of state transition functions

$$p_{i,j}(t) \equiv P\left[X(t) = j \middle| X(0) = i\right], i, j = 1,2,\ldots 195 \qquad (56)$$

for the continuous-time Markov chain can be solved from the forward Chapman-Kolmogorov equation [8]

$$\dot{P}(t) = P(t)Q, P(0) = I, P(t) = [p_{i,j}(t)], \qquad (57)$$

where Q is an infinitesimal generator of a rate transition matrix whose $(i,j)^{th}$ entry is given by the rate associated with the transition from current state *i* to next state *j*. The state probability mass function (pmf) at a time *t* is

$$\pi(t) = \left[\pi_1(t) \quad \pi_2(t) \quad \ldots \quad \pi_{195}(t)\right] \qquad (58)$$

and is calculated through

$$\pi(t) = \pi(0)P(t) \qquad (59)$$

This establishes a Markov model for the database unit seen in Figure 9.2.

9.2.2. Control Variables

The main objective of supervisory control is to reduce the adverse effects of server failure. To accomplish this, a combination of two control variables is instituted.

The control variable u_1 is used to dictate when the secondary server is allowed to serve queries when a failure elsewhere has occurred. u_1 is equal to *1* if the server whose primary sever serves the data class is down, or is restoring the same data class in another server. Otherwise, u_1 is equal to *0*.

The control variable u_2 is used to dictate which entity in queue takes precedence when the secondary server is allowed to serve. u_2 is equal to *1* when first query in queue takes precedence at an available server. Otherwise u_2 is equal to *0*, in which case the primary query takes precedence over the secondary query despite their orientation in the queue. As an example, server S_{AB} is given permission to serve both primary and secondary data classes. If the queue is ordered such that B precedes A (e.g. [0AB]), then A will be served before B is served. When u_2 =1, query B would be served first.

The control variable u_3 is used to dictate whether the database unit restores itself after failure. u_3 is equal to *1* when unit repair is allowed. Otherwise it is *0*.

9.2.3. Model augmentation to include delays with errors

Control action delays can occur at many state transitions. This study examines the case of a delay associated with server failure. In such a case, the action of server restoration is delayed.

The supervisory control of this system is state information based, thus, when a new state is entered that requires a

control action, information deficiency can lead to an incorrect control action.

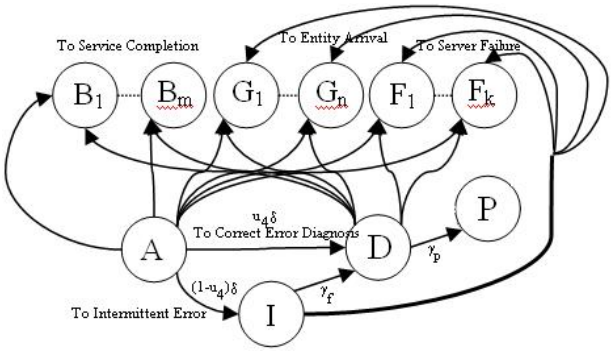


Figure 9.3 Control delay with decision error modeling

In Figure 9.3, let A be a state that is reached due to a server failure. Let P be the state that is reached when the primary data class of the server is restored. Let states $B_1,\ldots,B_m$ be the states entered upon service completion at operating servers. Let $G_1,\ldots,G_n$ be the states entered upon the arrival of a new query. Let $F_1,\ldots,F_k$ be the states entered upon the failure of another server. Let D be the state entered when a correct server failure is diagnosed. Let I represent the state entered if an wrong decision is made in transitioning out of state A.

In adding a delay and control error, 145 new states are created. It is assumed that there is no error or delay associated with restoring the secondary server after the primary has been restored. Since the system cannot immediately recognize server failure, in the period between physical failure and diagnosis, any queries entering the system that are allowed to be served by the failed server are sent to it with the belief that it is functioning. As a result the query is kept waiting until the server failure is diagnosed. Only upon correct diagnosis will queries be prevented from entering the server. Let δ be an exponentially distributed rate of control action delay, Figure 9.3 shows that a delay can lead to successful diagnosis of a failed server, resulting in the correct restoration of the failed unit, or an incorrect diagnosis of server failure, resulting in the attempted restoration of a functioning unit. The conditional probability of no decision error given that a server has failed is defined as u_4.

It was shown in [50] that the effect of delay in control actions and the effect decision errors always degrade the performance in terms of the transition probability to restoration of a failed server.

9.3. Performance analysis

In this section, performance measures in terms of mean time to system failure, steady-state system availability, expected query response time, and expected service overhead are reintroduced following the presentation in [50]. The performance is then evaluated for the model of the database system of Figure 9.2. The goal is to quantify the advantage of architecture in Figure 9.2 over that in Figure 9.1, and to understand the dependence of the performance on probability of no decision error, and the average delay. There was no clear advantage gained in the implementation of control variables u_1 and u_2. In this analysis, both values are set to one.

9.3.1. Mean time to system failure

By setting $u_3=0$, recovery to the initial state will not occur, and the system cannot leave any of the *40* failure (absorbing) states once it is entered. The other *300* states are transient states. The state probability vector can be represented as

$$\pi(t) \equiv [\underbrace{\pi_\tau(t)}_{1x300} \quad \underbrace{\pi_\alpha(t)}_{1x40}], \tag{60}$$

where vector π_τ contains the transient state probabilities and vector π_α contains the absorbing state probabilities. The rate transition matrix Q and the state transition function matrix $P(t)$ can be decomposed into

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ 0 & 0 \end{bmatrix}, P = \begin{bmatrix} P_{11}(t) & P_{12}(t) \\ 0 & 1 \end{bmatrix}. \tag{61}$$

From (57), (59), and (61), it can be found that the probability density function of time to system failure, or time to absorption, is

$$\dot{\pi}_\alpha(t) = \pi_\tau(0) P_{11}(t) Q_{12}, \pi_\alpha(0) = 0, \tag{62}$$

where

$$\pi_\tau(0) = [1 \quad 0 \quad \cdots \quad 0], P_{11}(t) = e^{Q_{11}t}. \tag{63}$$

Thus, the mean time to failure can be determined through the equation [24]

$$MTTF = -\pi_\tau(0) Q_{11}^{-1} I_\tau, I_\tau = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{64}$$

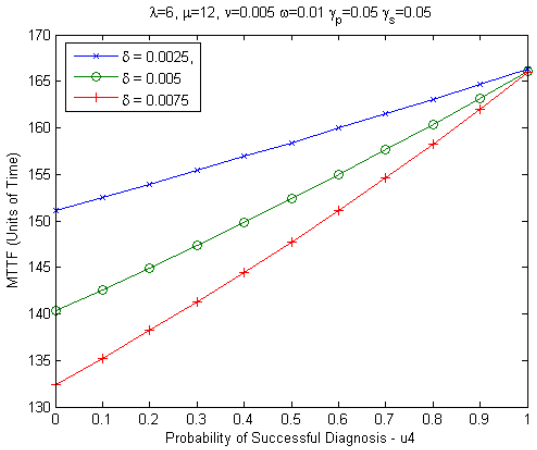


Figure 9.4 MTTF versus u_4

Figure 9. 4 shows the mean time to failure as a function of the conditional probability u_4 with the delay δ as a parameter. As the probability of successful error diagnosis increases, the mean time to failure also increases. The ability to successfully diagnose errors allows the unit to attend to failures more rapidly. This reduces the ability for another server to fail which results in unit to failure. Also, as the rate of delay increases, the mean time to failure decreases. A smaller delay makes the system easier to enter an intermittent error state. This is particular to the system setup. The

argument is that when less time is allowed to diagnose a failure, the likelihood of being able to identify correctly where the failure has occurred is reduced. As a result, the gain in less decision error outweighs the loss in longer delay in diagnosis process.

Figure 9.5 further observes the effects of delay, with restoration rate as the parameter. It can be seen that varying the success rate of failure diagnosis has a more profound effect on the mean time to failure than the restoration rate. For small values of delta, intermittant error states are reached more rapidly than other error states. As the delay time increases, the amount of time before an intermittant error state is reached increases as well. Thus, the mean time to failure increases.

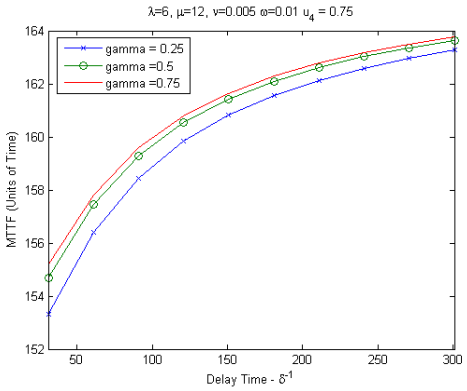


Figure 9.5 Unit MTTF versus delay time

9.3.2. *Steady State Availability*

The database unit is allowed to restore itself after it has failed by setting $u_3 = 1$. The unit is then restored with rate ω. The Markov chain for the system is irreducible, and therefore has a unique steady-state distribution [8]. The system's availability is the percentage of time when the system is not in a failure state, or

$$A_{sys} = 1 - \sum_{i=301}^{340} \pi_i(\infty) \tag{65}$$

where π_{301} *to* π_{340} can be solved through

$$\pi(\infty)Q = 0, \textit{ and } \sum_{x=1}^{340} \pi_x(\infty) \tag{66}$$

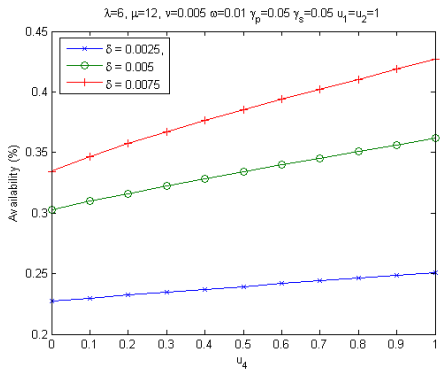


Figure 9.6 Steady state availability versus u_4

Figure 9.6. shows the steady state availability as a function of u_4 *with* the control delay as a parameter. An increase in the success of diagnosis results in greater system availability. However, for the delay rates chosen, the system has an unacceptably low availability. The faster the delay rate results in increased availability as well. A rapid failure diagnosis reduces the ability of other servers to fail.

Figure 9.7 shows the steady state availability as a function of control delay with restoration as a parameter. Restoration has little effect on reducing the effect of a prolonged delay on the system's availability. An increase in restoration rate only marginally increases the system's availability.

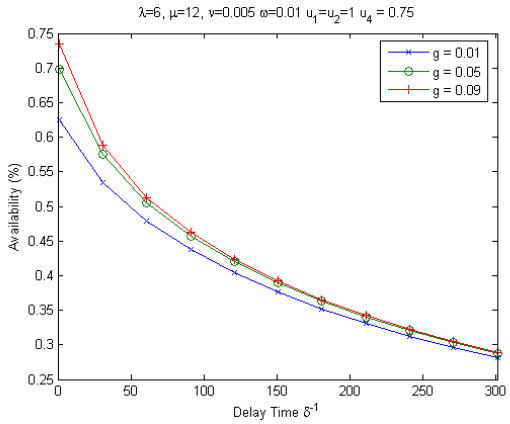


Figure 9.7 Steady state availability versus control delay

9.3.3. *Response Time*

The response time of the system is a measure of the amount of time a query resides in the database unit. It can be found by solving Little's Theorem $E[X] = \lambda_s E[R]$ [8], where $E[R]$ is the response time. The equation can be solved using

$$E[X] = \sum_{i=1}^{340} \pi_i(\infty) N_i \tag{67}$$

where N_i is the total number of queries in queue at state i. The steady state arrival is

$$\lambda_s = \sum_{i=1}^{340} \pi_i(\infty) \sum_{j=1}^{340} I_{ij} q_{ij} \tag{68}$$

where I_{ij} is an indicator function, valued as 0 if there is no transition from state i to j and 1 if there is a transition from state i to j, while q_{ij} is the corresponding value in the transition rate matrix, Q.

Figure 9.8 shows the response time as a function of u_4 with the delay rate as a parameter. As the success rate increases, the response time of the system decreases. Queries pass more rapidly through the system when server errors are properly diagnosed. Also, the less delay queries experience, the more rapidly they will pass through the system.

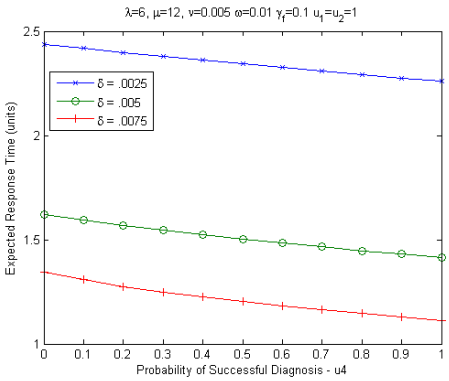


Figure 9.8 Response time versus u_4

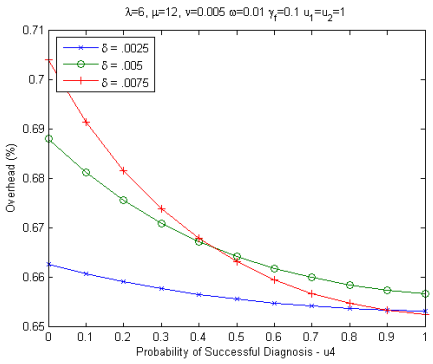


Figure 9.10 Overhead versus u_4

Figure 9.9 shows the response time as a function of control delay with restoration as a parameter. As the delay time increases, queries move more slowly through the system, increasing the response time. Also, reduced restoration rates of failed servers will also increase the time a query resides in the system.

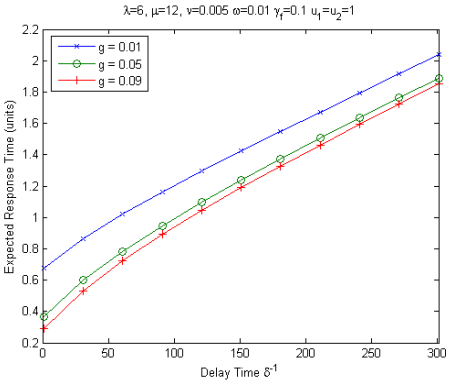


Figure 9.9 Response time versus control delay

9.3.4. Overhead

Overhead is a measure used to show the ratio of the time server invests on helping the database unit to survive longer to its overall busy time. It can be seen as

$$\theta = \frac{\Pr[S_{AB}\ restores/\ fails \mid unit\ has\ not\ \ failed]}{\Pr[S_{AB}\ restores/\ fails/\ serves \mid unit\ has\ not\ \ failed]} \quad (69)$$

Figure 9.10 shows the overhead results for the irreducible chain as a function of u_4 and uses the control delay as a parameter. All delay times display the same trend in overhead value. The more successful the system is in diagnosing errors, the less time servers will have to dedicate to restoring other servers.

Figure 9.11 shows overhead as a function of control delay with restoration as a parameter. An increase in delay time causes servers to dedicate themselves to the restoration of other servers and/or increases the possibility that those servers will themselves fail.

9.4. Section summary

The study of a single queue, multiple-class architecture did not prove to show improved performance over the multiple-queue architecture in the absence of delays with control errors. This is attributed to the low utilization of the system that houses only 3 queries, and to the routing control that partially offsets the disadvantage of the multiple queue architecture. The small number of queries is purposely chosen to make analytic modeling more tractable. It is expected that an increase in the number of queries will reveal the benefits of the single queue architecture. A simulation study will be conducted to validate the conjecture.

Analysis of the Markov chain shows that control action delays and decision errors can cripple the database unit. The benefits of allowing a server to serve a secondary class were marginal, and therefore, optimization of control policy matters little. Simulation studies with higher query traffic will better show if this whether control policy will matter more. In general the ability to minimize decision error was more important than ensuring rapid restoration.

A most interesting future research area would be to evaluate decision errors and control delays when more detailed model for decision and control processes are established.

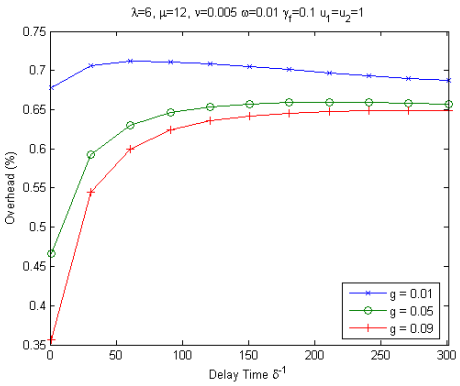


Figure 9.11 Overhead versus control delay

10. Simulation of the Single Queue Database Unit & Comparison with the Multi-queue Unit

10.1. Problem description

The above section developed a single queue Markov model of a database unit. It compared that architecture with a multiple queue database unit architecture devised by Wu and Metzler using the performance measures, mean time to failure, system availability, expected response time, and system overhead. The single queue system improved in expected response time and in system overhead as compared with the multiple queue system. However, the improvement observed was less than expected. The above attempt failed to consider the effect that the varied arrival rates have on the system. It is expected that varying the arrival rate will prove the single queue's advantage over the multiple queue more definitively.

The single queue model allows service by the secondary class server only when the corresponding primary server is in a failed or restoring state. The Markov model is redefined to allow the secondary class server to serve at any time when it is not otherwise serving, failed, or restoring. The new model contains 153 viable states. The original model contained 162 viable states. The change in viable states causes an improvement in expected response time, system overhead, and server utilization.

This section will numerically analyze the effect of increasing the arrival rate by 3.33 times the nominal value for the closed system. It will also compare the multiple queue system configuration and single queue configuration through simulation. Numerical analysis will be performed on the single queue system modified to allow the secondary server to serve at any time it is not busy, failed, or restoring. The single queue system modified to allow the secondary server to serve will then be analyzed through simulation. All system parameters are considered to be: λ=6, $\gamma_p = \gamma_s = 0.05$, ν = 0.005, μ = 12, ω = 0.01, $u_1 = u_2 = u_3$ =1 unless otherwise noted.

10.2. Original System – Numerical Analysis

Figure 10.1 shows the effect of increasing the arrival rate of the closed single queue system on overhead. The system is closed, therefore only three queries are allowed in the system at any given moment. By reducing the amount of time spent in the process delay, the faster they are returned to the queue. This results in a reduction in the time a server is idle and an increase in the time that the server is busy. At ν = 0.001, there is an 8% reduction in overhead. At ν = 0.01 there is a 10% reduction in overhead. As the failure rate increases, servers fail more often. Therefore other servers spend an increased amount of time repairing failed servers. This results in a convergence of the two overhead plots.

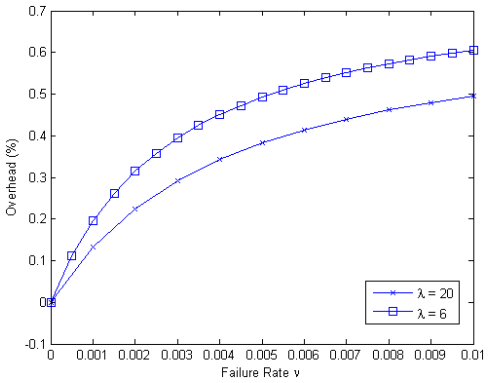


Figure 10.1: Single queue system overhead versus failure rate with varied arrival rate

Figure 10.2 shows the effect of increasing the arrival rate of the closed single queue system on response time. A significant increase in response time is observed. At γ = 0, there is a 1.2 time unit difference in response time. At γ = 0.1, there is a 0.42 time unit difference in response time. The increase in response time is due to the effect of the restoration rate relative to service rate. Queries are serviced with a rate of 12 queries/unit time. By allowing rapid service followed by a short arrival delay, queries pass rapidly through the system. However, in the event of a failure, the restoration rate is varied only between zero and 0.1, two orders of magnitude lower than the parameters μ=12 and λ = 20. Therefore, queries that cannot be served must wait in the queue. When λ = 6, these queries waited for less time in the queue because they spent more time in the delay element. When λ = 20, the queries waited for more time in the queue because they spent less time in the delay element. Also, as the restoration rate increases, the difference in response time decreases because servers are restored more quickly and queries wait in the queue for a lower amount of time.

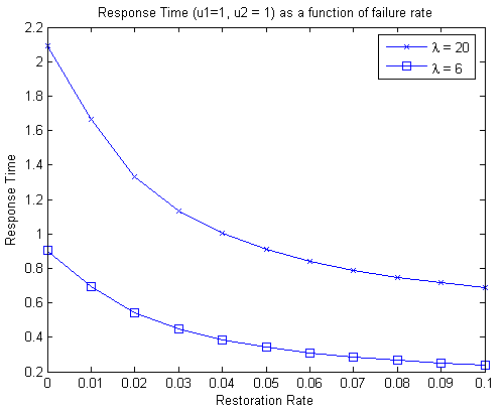


Figure 10.2: Single queue system response time versus failure rate with varied arrival rate

10.3. Original System – Arena Simulation

Figure 10.3 compares the response times of the single queue system and the multiple queue system proposed by Wu and Metzler as a function of arrival rate. The single queue system performs better than the multiple queue

system. When the arrival rate is 2, the single queue system outperforms the multiple queue system by 4.2 time units. When the arrival rate is 8, the single queue system outperforms the multiple queue system by 6.18. When the arrival rate is 16, the single queue system outperforms the multiple queue system by 14.1. Small arrival rates indicate that few queries are entering the system. Thus, the advantage of the single queue system is less apparent. Greater arrival rates indicate more queries are entering system. As the value of λ increases, the greater the advantage of the single queue system has in terms of response time.

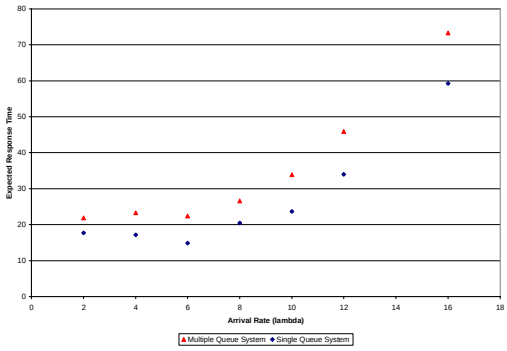


Figure 10.3: Expected response times versus arrival rate

Figure 10.4 compares the overhead of the single queue system and the multiple queue system proposed by Wu and Metzler as a function of failure rate. The single queue system performs better than the multiple queue system. When the arrival rate is 6, the single queue system outperforms the multiple queue system by 5%. When the arrival rate is 12, the single queue system outperforms the multiple queue system by 6.6%. When the arrival rate is 16, the single queue system outperforms the multiple queue system by 9.81%. As the value of λ increases, the greater the advantage of the single queue system has in terms of overhead.

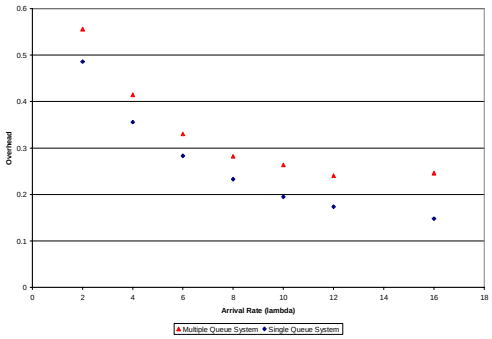


Figure 10.4: Expected response times versus arrival rate

10.4. System where secondary server is allowed to serve at all times

Nominal arrival rate, λ=6

In the original system the secondary server was used primarily to restore failed primary servers. However, it was given the ability to serve customers under special conditions. Now the secondary server is allowed to serve whenever it is able. Conditions where the secondary cannot serve occur when: it is busy, it is failed, it is repairing, or the corresponding primary is idle. The model for this case will be called the 'New System'. The previously discussed system will be called the, 'Original System'.

Figure 10.5 shows the effect of increasing the failure rate of the closed single queue system on overhead. This system shows that the new system has a higher overhead than the original system. This is an unexpected result. By increasing the number of servers a query can choose, it is expected to result in a lower overhead. However, the closed system only allows three queries in the system at any time. Allowing queries to choose multiple servers causes the idle time of each server to increase. If the idle time increases more than the busy time for each server the overhead will increase. The result is demonstrated in figure 5. A decrease in response time and an increase in utilization would support this hypothesis.

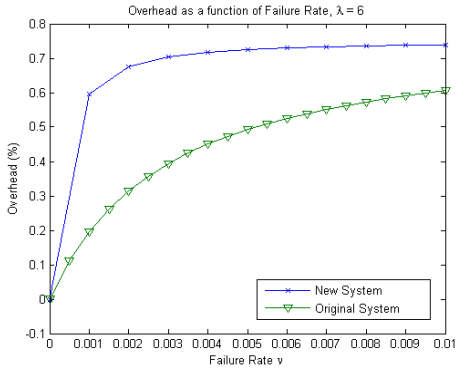


Figure 10.5: Original single queue system overhead and new single queue system overhead versus failure rate

Figure 10.6 shows the effect of increasing the restoration rate of the closed single queue system on response time. The trend in response time as a function of restoration rate is expected. The new system has a lower response time then the original system. Queries spend less time in the new system than in the original system. On average, the new system has a 9.93% lower response time. By allowing the secondary server to serve whenever it is able, queries are served faster and spend 9.93% less time in the system.

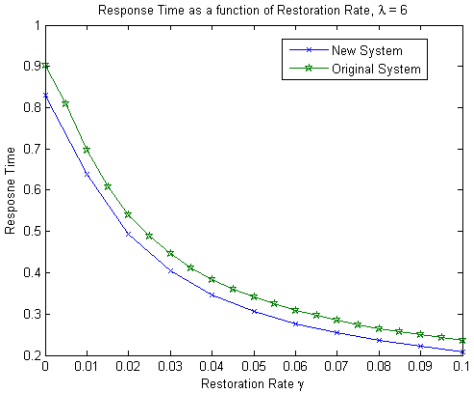


Figure 10.6: Original single queue system response time and new single queue system response time versus restoration rate

Figure 10.7 shows the effect of increasing the restoration rate of the closed single queue system on server utilization. The new system shows a dramatic increase in the utilization of a server. On the average, the new system is utilized 50.54 % more than the original system. When the secondary class server is allowed to serve queries, a query's waiting time in the queue is decreased. In the event of no server error, all servers are utilized as long as one query is requesting a different class type than the other queries. For example, if three queries are in the system and they are requesting class A, A, and B data. The A queries can be served simultaneously by S_{AB}, and S_{CA} while the B query is served by S_{BC}. In the original system S_{CA} would not provide service and the second class A query would wait in queue.

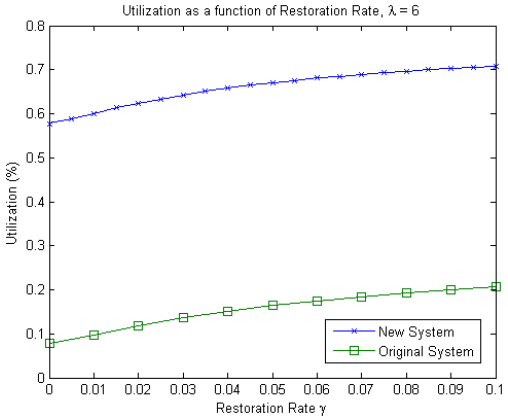


Figure 10.7: Original single queue system utilization and new single queue system utilization versus restoration rate

Nominal arrival rate, λ=20

Next, an increase of the query arrival rate is implemented to further analyze the new system. Figure 8 shows the effect of increasing the query arrival rate on the new system's overhead. There is a 14% reduction in overhead for the new system to when the arrival rate was 6. However, the overhead remains greater than the original system at the nominal arrival rate. When the arrival rate was 6, servers became better utilized; however the limited number of queries caused an increase in server idle time. The result was an increase in the utilization compared to the original system. In this case, increasing the arrival rate of queries causes a decrease in idle time because queries are returned back to the system more rapidly. This increase is not enough to equalize the overhead value to that of the original system.

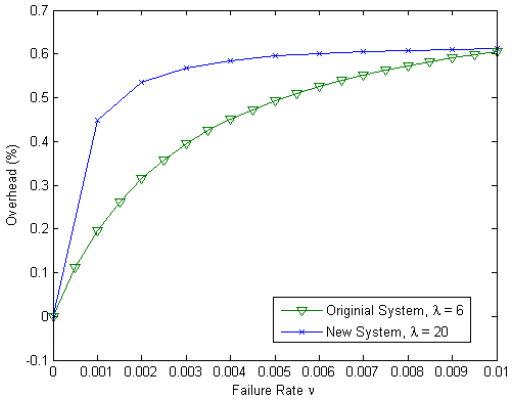


Figure 10.8: Original single queue system overhead and new single queue system (λ=20) overhead versus failure rate

Figure 10.9 shows the effect of increasing the query arrival rate on the new system's response time. A significant increase in response time is observed. At γ = 0, there is a 0.77 time unit difference in response time. At γ = 0.1, there is a 0.3 time unit difference in response time. This difference is less than that observed in figure 2. This indicates that the new system improves the system response time compared to the original system when λ=20. The increase in response time is due to the effect of restoration rate relative to service rate. Queries are serviced with a rate of 12 queries/unit time. By allowing rapid service followed by a short arrival delay, queries pass rapidly through the system However, in the event of a failure, the restoration rate is only varied between zero and 0.1, two orders of magnitude lower than parameters μ=12 and λ = 20. Therefore, queries that cannot be served must wait in the queue. When λ = 6, these queries waited in the queue for less time because they spent more time in the delay element. When λ = 20, the queries waited for more time in the queue because they spent less time in the delay element. Also, as the restoration rate increases, the difference in response time decreases because servers are restored more quickly and queries wait in the queue for a shorter amount of time.

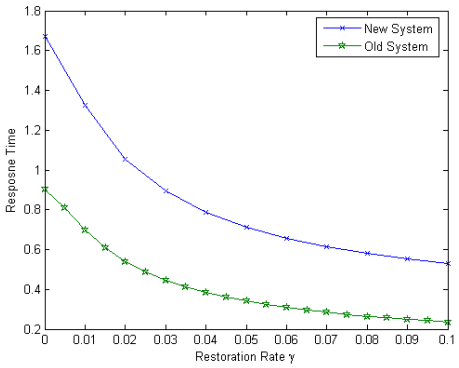


Figure 10.9: Original single queue system response time (λ=6) and new single queue system (λ=20) response time versus restoration rate

Figure 10.10 shows the effect of increasing the query arrival rate on the new system's server utilization. The new system shows a dramatic increase in the utilization of a server. On the average, the new system is utilized 53.25 % more than the original system. This is an increase in server utilization compared to when λ=6 for the new system. When the secondary class server is allowed to serve queries, a query's waiting time in the queue is decreased. In the event of no server errors, all servers are utilized as long as one query is requesting a different class type than the other queries. For example, if three queries are in the system and they are requesting class A, A, and B data. The A queries can be served simultaneously by S_{AB}, and S_{CA} while the B query is served by S_{BC}. In the original system S_{CA} would not provide service and the second class A query would wait in queue.

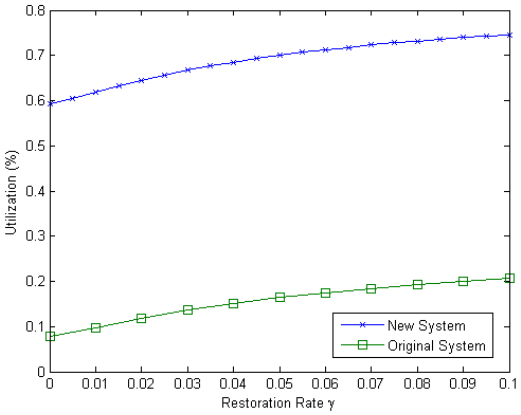


Figure 10.10: Original single queue system utilization and new single queue system (λ=20) utilization versus restoration rate

10.5. New System – Simulation

Figure 10.11 shows a comparison of the response time of the new and original systems as a function of arrival rate. At a low arrival rate of λ = 2 there is a 2.8 time unit reduction in response time. At a higher arrival rate of λ = 16, this reduction is 4.95 time units. Queries pass through more rapidly in the new system. At low arrival rates this is not as apparent because the traffic is not heavy. As the arrival rate increases, more queries are present in the system and the advantage becomes more apparent. The improvement of the new system over the original system is not as great as the original single queue system over the multiple queue system.

Figure 10.12 shows a comparison of the overhead of the new and original systems as a function of failure rate. The new system results in higher overhead values for all measured arrival rates. At a low arrival rate of λ = 2 there is a 4.4% increase in overhead. At a higher arrival rate of λ = 16, there is a decrease of 0.7%. Overhead is a measure of the percentage of time a server spends in a failed/restoring state with respect to the time the system spends in a failed/restoring/serving state. Low arrival rates result in a higher overhead for the 'new' system. The 'new' system decreases the idle time of each server by allowing them to serve more query types. However, the service time does not increase as much as the idle time decreases. This results in an increase in overhead. Queries back up in the queue when the query arrival rate becomes greater than the query service rate. This results in a consistently busy server regardless of the query class being served. Thus, the overhead rates converge to the same value.

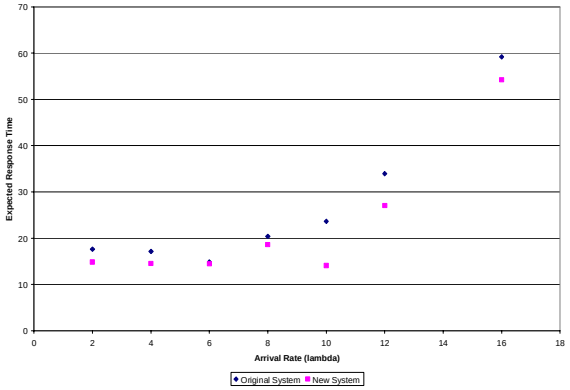


Figure 10.11: Simulated expected response time for original system and new system as a function of arrival rate

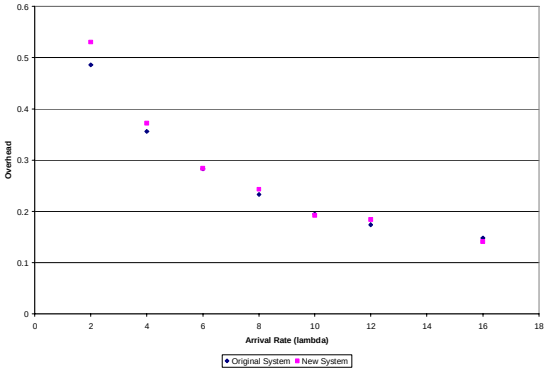


Figure 10.12: Simulated expected response time for original system and new system as a function of arrival rate

10.6. Section summary

The single queue system has been shown to perform better than the multiple queue system. An increase in the arrival rate of queries results in an increased advantage of the single queue system over the multiple queue system with respect to expected response time and system overhead. Altering the single queue system to increase service by the secondary class server results in an improved response time performance. This improvement is less than the improvement of the original single queue system over the multiple queue system. Overhead does not improve considerably because of the increase in the number of queries in the system.

REFERENCES

[1] T. Altiok, C. Vu Doy, and M. Baykal-Gursoy, Two load charing processors with failures, *Computers Ops Res.*, vol.35, pp183-189, 1998.

[2] P.J. Antsaklis, Guest Editor, Hybrid Systems: Theory and Applications, Special Issue *Proceedings of the IEEE*, vol.88, pp.879-1130, 2000.

[3] J. Banks, J.S. Carson, II, B.L. Nelson and D. M. Nicol, "*Discrete – Event System Simulation*", New Jersey: Prentice – Hall Inc, 2001.

[4] Bellman, R. (1957). *Dynamic Programming*, Princeton University Press.

[5] Bhardwaj, M., and A.P. Chandrakasan, Bounding the lifetime of sensor networks via optimal role assignments. *Proc. IEEE Infocom*, 2002.

[6] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi, "*Queuing Networks and Markov Chains*", John Weiley & Sons, 1998.

[7] G. Casella, and R. L. Berger, *Statical Inference*, 2nd Edition, Duxbury, 2002.

[8] C.G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer, 1999.

[9] J. M. Carlson, and J. Doyle, Highly optimized tolerance: robustness and design in complex systems, *Physical Review Letters* vol.84, pp.2529-2532, 2000.

[10] J. B. Cruz, Jr., M. A. Simaan, A. Gacic, H. Jiang, B. Letellier, M. Li, and Yong Liu, Game-theoretic modeling and control of a military air operation, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, pp.1393-1405, 2001.

[11] M. Curry, J. Wohletz, C.G. Cassandras, and D. Castanon, Modeling and control of a joint air operation environment with imperfect information, *Proc. SPIE 16th Annual Intl. Symposium*, vol.4716, pp.41-52, 2002.

[12] P. K. Davis, Exploratory analysis and a case history of multi-resolution multi-perspective modeling, *Proc. SPIE 14th Annual Intl. Symposium*, vol.4026, pp.2-15, 2000.

[13] S. De, and C. Qiao, Does packet replication along multipath really help, *Proc. ICC*, 2003.

[14] Draper Laboratory, *Closed-Loop, Hierarchical Control of Military Air Operations*, Final Report CSDL-R-2914 to DARPA Joint Forces Air Component Commander Program, 2001.

[15] C.E. Ebeling, *An introduction to Reliability and Maintainability Engineering*, McGraw-Hill, 1997.

[16] D. Ganesan, R. Govinden, S. Shenker, and D. Estrin, Highly-resilient, energy-efficient multipath routing in wireless sensor networks, *ACM Mobile Computing and Communications Review*, Vol. 1, Issue 2, 2002.

[17] D. Ghose, J.L. Speyer, and J.S. Shamma, A game theoretical multiple resource interaction approach to temporal resource allocation in an air campaign, Annals of Operations Research, pp.15-40, 2002.

[18] P.G. Harrison, N.M. Patel, and E. Pitel, Reliability modeling using G-queues, *European Journal of Operational Research*, vol.126, pp.273-287, 2000.

[19] G. Hoblos, M. Staroswiecki and A. Aitouche, Optimal design of fault tolerant sensor networks, Proc. *IEEE International Conference on Control Applications*, 2000.

[20] R.A. Howard, *Dynamic probabilistic systems: V1, Markov Models, V2, Semi-Markov and Decision Processes*, Wiley, 1971.

[21] J. Jelinek, Models for computing combat risk, *Proc. SPIE 16th Annual Intl. Symposium*, vol.4716, pp.52-63, 2002.

[22] E.P.C. Kao, *An Introduction to Stochastic Processes*, Duxbury Press, 1997.

[23] W. D. Kelton, R. P. Sadowski and D. T. Sturrock, *Simulation with Arena*, 3rd Ed., McGraw-Hill, 2004.

[24] L. Kleinrock, *Queuing Systems Volume II: Computer Applications*, John Wiley & Sons, 1976.

[25] P.R. Kumar, New technological vistas for systems and control, *IEEE Control Systems Magazine*, vol. pp. 24-37, 2001.

[26] J.N. Laneman and G. W. Wornell, Distributed space-time-coded protocols for exploiting cooperative diversity in wireless networks, *IEEE Transactions on Information Theory*, Vol. 49, pp. 2415-2425, 2003.

[27] A.M. Law and W.D. Kelton, "*Simulation Modeling and Analysis*", 3rd ed., McGraw Hill, 2000.

[28] X. Li, "Energy efficient wireless sensor networks with transmission diversity", *Electronics Letters*, **39**, pp.1753-1755, 2003.

[29] X. Li, "Space-time coded multi-transmission among distributed transmitter without perfect synchronization", *IEEE Signal Processing Letter*, **11**, 2004.

[30] X. Li, and N.E. Wu. "Power efficient wireless sensor networks with distributed transmission-induced space spreading", *Proc. 37th Asilomar Conf. Signals., Syst., Comput.* ,2003.

[31] D.Q. Mayne and H. Michalska, Receding horizon control of nonlinear systems, *IEEE Trans. Automatic Control*, Vol. 35, pp.814–824, 1990.

[32] J.M. Metzler, and N.E. Wu "A Simulation Study of the Effect of Supervisory Control on a Redundant Database Unit," Report to AFRL, 2005..

[33] F. Ordonez and B. Krishnamachari, Optimal information extraction in energy-limited wireless sensor networks, *IEEE Journal on Selected Areas in Communications, Special Issue on Fundamental Performance Limits of Wireless Sensor Networks*, 2004.

[34] T.S. Rappaport, Wireless Communications, Principle and Practice}, 2nd ed., Prentice Hall, 2002.

[35] Rockwell Software, Inc. *Arena*, Academic Version 7.01.00, 2003.

[36] A.E. Sendonaris, Erkip and B. Aazhang, User cooperation diversity, Part I, II, *IEEE Transactions on Communications*, Vol. 51, pp.1927-1948, 2003.

[37] D. Thorsley and D. Teneketzis, "Diagnosability of stochastic discrete event systems," *IEEE Trans. Automatic Control*, 50(4), pp. 476-492, 2005.

[38] D. Torrieri, Calculation of node-pair reliability in large networks with unreliable nodes, *IEEE Transactions on Reliability*, Vol. 43, pp.375-377, 1994.

[39] K.S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, Prentice-Hall, 1982.

[40] K.H. Wang and J.C. Ke, Probabilistic analysis of a repairable system with warm standbys plus balking and reneging, Applied Mathematical Modeling}, vol.27, pp.327-336, 2003.

[41] J.M. Wohletz, D.A. Castanon, M.L. Curry, Closed-loop control for joint air operations, *Proc. American Control Conference*, pp.4699-4704, 2001.

[42] N.E. Wu, "Coverage in fault tolerant control," *Automatica*, vol.40, pp.537-548, 2004.

[43] N.E. Wu, Reliability of fault tolerant control systems, part i \& ii, *Proc. IEEE Conference on Decision and Control*, 2001.

[44] N. E. Wu and T. Busch, "An example of supervisory control in C2," *Proc. IEEE Conference on Decision and Control*, 2004.

[45] N. E. Wu, and T. Busch, "Operational reconfigurability in command and control," *Proc. American Control Conference*, 2004.

[46] N.E. Wu, and T. Busch, Strategic reconfigurability in air operations, *Proc. IEEE Conference on Decision and Control*, 2003.

[47] N.E. Wu, X. Li and T. Busch, "Reliability and Feedback in Multiple-Hop Wireless Network", *IFAC World Congress*, 2005.

[48] N.E. Wu, and S. Thavamani, Simulation of a reconfigurable C2 system using Arena, submitted to *IEEE Conference on Decision and Control*, 2004.

[49] N.E. Wu, and J. M. Metzler, "Reconfigurability in command and control systems: data loss prevention in a redundant database unit," *Report* to *AFRL RRS*, 2005.

[50] N.E. Wu, J.M. Metzler, and M. Linderman, "Performance of a Controlled Database Unit Subject to Decision Errors and Control Delays", *Proc. 8th International Workshop on Discrete Event System*, 2006.

[51] N.E. Wu, J.M. Metzler, and M. Linderman, "Supervisory Control of a Database Unit", *Proc. IEEE Conference on Decision and Control*, 2005.

[52] N.E. Wu, K. Zhou, and G. Salomon, Control reconfigurability of linear time-invariant systems, *Automatica*, vol.36, pp.1767-1771, 2000.

[53] F. Xue, and P. R. Kumar, The number of neighbors needed for connectivity of wireless networks, *ACM Journal of Wireless Networks*, Vol. 10, pp.169-181, 2004.

[54] S. Zacks, *Introduction to Reliability Analysis: Probability Models and Statistics Methods*, Springer-Verlag, 1992.

Appendix

Table 7.3 Transitions and transition rates of the database unit model with all rates valid for all policies, based on which matrix Q is formed

	State	States						Arrivals						Completions						Failures											Restorations			
	#	Q_{AB}	Q_{BC}	Q_{CA}	S_{AB}	S_{BC}	S_{CA}	λ_1	R	λ_2	R	λ_3	R	μ_p			R	μ_p	R	v			R	ReRoute		R	System Failure			R	γ_p	R	γ_s	R
	1	0	0	0	0	0	0	2	h	3	h	4	h	x	x	x	x	x	x	14	15	16	v	x	x	x	x	x	x	x	x	x	x	x
Ones	2	1	0	0	0	0	0	11	2/3*h	8	2/3*h	9	2/3*h	1	x	x	mp	x	x	36	37	38	(u1*n2+u1*u2)*v	37	38	n1*u2*v	142	37	38	n1*n2*v	x	x	x	x
	3	0	1	0	0	0	0	8	2/3*h	12	2/3*h	10	2/3*h	1	x	x	mp	x	x	39	40	41	(u1*n2+u1*u2)*v	15	41	n1*u2*v	39	142	41	n1*n2*v	x	x	x	x
	4	0	0	1	0	0	0	9	2/3*h	10	2/3*h	13	2/3*h	1	x	x	mp	x	x	42	43	44	(u1*n2+u1*u2)*v	43	16	n1*u2*v	42	43	142	n1*n2*v	x	x	x	x
	5	0	0	0	1	0	0	17	(u1*n2*1/3 + u1*u2*1/6)*h	20	(u1*n2*1/3 + u1*u2*1/6)*h	23	(u1*n2*1/3 + u1*u2*2/3)*h	x	x	x	x	x	x	141	x	x	2*v	x	x	x	x	x	x	x	x	x	1	u1*gs
	6	0	0	0	0	1	0	18	(u1*n2*1/3 + u1*u2*2/3)*h	21	(u1*n2*1/3 + u1*u2*1/6)*h	24	(u1*n2*1/3 + u1*u2*1/6)*h	x	x	x	x	x	x	141	x	x	2*v	x	x	x	x	x	x	x	x	x	1	u1*gs
	7	0	0	0	0	0	1	19	(u1*n2*1/3 + u1*u2*1/6)*h	22	(u1*n2*1/3 + u1*u2*2/3)*h	25	(u1*n2*1/3 + u1*u2*1/6)*h	x	x	x	x	x	x	141	x	x	2*v	x	x	x	x	x	x	x	x	x	1	u1*gs
Twos	8	1	1	0	0	0	0	50	1/3*h	48	1/3*h	26	1/3*h	2	3	x	mp	x	x	66	67	68	(u1*n2+u1*u2)*v	37	68	n1*u2*v	143	143	68	n1*n2*v	x	x	x	x
	9	1	0	1	0	0	0	49	1/3*h	26	1/3*h	47	1/3*h	2	4	x	mp	x	x	72	73	74	(u1*n2+u1*u2)*v	73	38	n1*u2*v	143	73	143	n1*n2*v	x	x	x	x
	10	0	1	1	0	0	0	26	1/3*h	46	1/3*h	45	1/3*h	3	4	x	mp	x	x	69	70	71	(u1*n2+u1*u2)*v	43	41	n1*u2*v	69	143	143	n1*n2*v	x	x	x	x
	11	2	0	0	0	0	0	60	1/3*h	50	1/3*h	49	1/3*h	2	x	x	mp	x	x	93	94	95	(u1*n2+u1*u2)*v	94	95	n1*u2*v	145	94	95	n1*n2*v	x	x	x	x
	12	0	2	0	0	0	0	48	1/3*h	61	1/3*h	46	1/3*h	3	x	x	mp	x	x	96	97	98	(u1*n2+u1*u2)*v	15	98	n1*u2*v	96	145	98	n1*n2*v	x	x	x	x
	13	0	0	2	0	0	0	47	1/3*h	45	1/3*h	62	1/3*h	4	x	x	mp	x	x	99	100	101	(u1*n2+u1*u2)*v	100	16	n1*u2*v	99	100	145	n1*n2*v	x	x	x	x
	14	0	0	0	2	0	0	36	(u1*n2*1/3)*h	39	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*h	42	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*h	x	x	x	x	x	x	141	x	x	2*v	x	x	x	x	x	x	x	5	u1*gp	x	x
	15	0	0	0	0	2	0	37	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*h	40	(u1*n2*1/3)*h	43	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*h	x	x	x	x	x	x	141	x	x	2*v	x	x	x	x	x	x	x	6	u1*gp	x	x
	16	0	0	0	0	0	2	38	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*h	41	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*h	44	(u1*n2*1/3)*h	x	x	x	x	x	x	141	x	x	2*v	x	x	x	x	x	x	x	7	u1*gp	x	x
	17	1	0	0	1	0	0	51	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	27	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	30	(u1*n2*1/3 + u1*u2*2/3)*2/3*h	x	x	x	x	x	x	142	x	x	2*v	x	x	x	x	x	x	x	x	x	2	u1*gs
	18	1	0	0	0	1	0	52	(u1*n2*1/3 + u1*u2*2/3)*2/3*h	28	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	31	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	6	x	x	mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	x	x	2	u1*gs
	19	1	0	0	0	0	1	53	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	29	(u1*n2*1/3 + u1*u2*2/3)*2/3*h	32	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	7	x	x	n1*mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	x	x	2	u1*gs
	20	0	1	0	1	0	0	27	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	54	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	33	(u1*n2*1/3 + u1*u2*2/3)*2/3*h	5	x	x	n1*mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	x	x	3	u1*gs
	21	0	1	0	0	1	0	28	(u1*n2*1/3 + u1*u2*2/3)*2/3*h	55	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	34	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	x	x	x	x	x	x	142	x	x	2*v	x	x	x	x	x	x	x	x	x	3	u1*gs
	22	0	1	0	0	0	1	29	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	56	(u1*n2*1/3 + u1*u2*2/3)*2/3*h	35	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	7	x	x	mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	x	x	3	u1*gs
	23	0	0	1	1	0	0	30	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	33	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	57	(u1*n2*1/3 + u1*u2*2/3)*2/3*h	5	x	x	mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	x	x	4	u1*gs
	24	0	0	1	0	1	0	31	(u1*n2*1/3 + u1*u2*2/3)*2/3*h	34	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	58	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	6	x	x	n1*mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	x	x	4	u1*gs
	25	0	0	1	0	0	1	32	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	35	(u1*n2*1/3 + u1*u2*2/3)*2/3*h	59	(u1*n2*1/3 + u1*u2*1/6)*2/3*h	x	x	x	x	x	x	142	x	x	2*v	x	x	x	x	x	x	x	x	x	4	u1*gs
Threes	26	1	1	1	0	0	0	x	0	x	0	x	0	8	9	10	mp	x	x	111	112	113	(u1*n2+u1*u2)*v	73	68	n1*u2*v	144	144	144	n1*n2*v	x	x	x	x
	27	1	1	0	1	0	0	75	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	87	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	63	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	17	x	x	n1*mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	x	x	8	u1*gs
	28	1	1	0	0	1	0	76	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	88	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	64	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	21	x	x	mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	x	x	8	u1*gs
	29	1	1	0	0	0	1	77	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	89	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	65	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	22	19	x	n1*mp	19	u1*mp	143	x	x	2*v	x	x	x	x	x	x	x	x	x	8	u1*gs
	30	1	0	1	1	0	0	78	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	63	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	90	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	17	x	x	mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	x	x	9	u1*gs
	31	1	0	1	0	1	0	79	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	64	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	91	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	24	18	x	n1*mp	24	u1*mp	143	x	x	2*v	x	x	x	x	x	x	x	x	x	9	u1*gs
	32	1	0	1	0	0	1	80	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	65	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	92	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	25	x	x	n1*mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	x	x	9	u1*gs
	33	0	1	1	1	0	0	63	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	81	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	84	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	23	20	x	n1*mp	20	u1*mp	143	x	x	2*v	x	x	x	x	x	x	x	x	x	10	u1*gs
	34	0	1	1	0	1	0	64	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	82	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	85	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	21	x	x	n1*mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	x	x	10	u1*gs
	35	0	1	1	0	0	1	65	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	83	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	86	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	25	x	x	mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	x	x	10	u1*gs
	36	1	0	0	2	0	0	93	(u1*n2*1/3)*2/3*h	66	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*2/3*h	72	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*2/3*h	x	x	x	x	x	x	142	x	x	2*v	x	x	x	x	x	x	x	17	u1*gp	x	x
	37	1	0	0	0	2	0	94	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*2/3*h	67	(u1*n2*1/3)*2/3*h	73	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*2/3*h	15	x	x	n1*mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	18	u1*gp	x	x
	38	1	0	0	0	0	2	95	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*2/3*h	68	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*2/3*h	74	(u1*n2*1/3)*2/3*h	16	x	x	mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	19	u1*gp	x	x
	39	0	1	0	2	0	0	66	(u1*n2*1/3)*2/3*h	96	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*2/3*h	69	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*2/3*h	14	x	x	mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	20	u1*gp	x	x
	40	0	1	0	0	2	0	67	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*2/3*h	97	(u1*n2*1/3)*2/3*h	70	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*2/3*h	x	x	x	x	x	x	142	x	x	2*v	x	x	x	x	x	x	x	21	u1*gp	x	x
	41	0	1	0	0	0	2	68	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*2/3*h	98	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*2/3*h	71	(u1*n2*1/3)*2/3*h	16	x	x	n1*mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	22	u1*gp	x	x
	42	0	0	1	2	0	0	72	(u1*n2*1/3)*2/3*h	69	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*2/3*h	99	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*2/3*h	14	x	x	n1*mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	23	u1*gp	x	x
	43	0	0	1	0	2	0	73	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*2/3*h	70	(u1*n2*1/3)*2/3*h	100	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*2/3*h	15	x	x	mp	x	x	142	x	x	2*v	x	x	x	x	x	x	x	24	u1*gp	x	x
	44	0	0	1	0	0	2	74	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*2/3*h	71	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*2/3*h	101	(u1*n2*1/3)*2/3*h	x	x	x	x	x	x	142	x	x	2*v	x	x	x	x	x	x	x	25	u1*gp	x	x
	45	0	1	2	0	0	0	x	0	x	0	x	0	13	10	x	mp	x	x	114	120	126	(u1*n2+u1*u2)*v	100	41	n1*u2*v	114	146	146	n1*n2*v	x	x	x	x
	46	0	2	1	0	0	0	x	0	x	0	x	0	10	12	x	mp	x	x	115	121	127	(u1*n2+u1*u2)*v	43	98	n1*u2*v	115	146	146	n1*n2*v	x	x	x	x
	47	1	0	2	0	0	0	x	0	x	0	x	0	13	9	x	mp	x	x	116	122	128	(u1*n2+u1*u2)*v	122	38	n1*u2*v	146	122	146	n1*n2*v	x	x	x	x
	48	1	2	0	0	0	0	x	0	x	0	x	0	12	8	x	mp	x	x	117	123	129	(u1*n2+u1*u2)*v	37	129	n1*u2*v	146	146	129	n1*n2*v	x	x	x	x
	49	2	0	1	0	0	0	x	0	x	0	x	0	9	11	x	mp	x	x	118	124	130	(u1*n2+u1*u2)*v	124	95	n1*u2*v	146	124	146	n1*n2*v	x	x	x	x
	50	2	1	0	0	0	0	x	0	x	0	x	0	8	11	x	mp	x	x	119	125	131	(u1*n2+u1*u2)*v	94	131	n1*u2*v	146	146	131	n1*n2*v	x	x	x	x
	51	2	0	0	1	0	0	102	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	75	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	78	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	x	x	x	x	x	x	145	x	x	2*v	x	x	x	x	x	x	x	x	x	11	u1*gs
	52	2	0	0	0	1	0	103	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	76	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	79	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	18	x	x	mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	x	x	11	u1*gs
	53	2	0	0	0	0	1	104	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	77	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	80	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	19	x	x	n1*mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	x	x	11	u1*gs
	54	0	2	0	1	0	0	87	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	105	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	81	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	20	x	x	n1*mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	x	x	12	u1*gs
	55	0	2	0	0	1	0	88	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	106	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	82	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	x	x	x	x	x	x	145	x	x	2*v	x	x	x	x	x	x	x	x	x	12	u1*gs
	56	0	2	0	0	0	1	89	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	107	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	83	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	22	x	x	mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	x	x	12	u1*gs
	57	0	0	2	1	0	0	90	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	84	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	108	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	23	x	x	mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	x	x	13	u1*gs
	58	0	0	2	0	1	0	91	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	85	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	109	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	24	x	x	n1*mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	x	x	13	u1*gs
	59	0	0	2	0	0	1	92	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	86	(u1*n2*1/3 + u1*u2*2/3)*1/3*h	110	(u1*n2*1/3 + u1*u2*1/6)*1/3*h	x	x	x	x	x	x	145	x	x	2*v	x	x	x	x	x	x	x	x	x	13	u1*gs
	60	3	0	0	0	0	0	x	0	x	0	x	0	11	x	x	mp	x	x	132	135	138	(u1*n2+u1*u2)*v	135	138	n1*u2*v	147	135	138	n1*n2*v	x	x	x	x
	61	0	3	0	0	0	0	x	0	x	0	x	0	12	x	x	mp	x	x	133	136	139	(u1*n2+u1*u2)*v	15	139	n1*u2*v	133	147	139	n1*n2*v	x	x	x	x
	62	0	0	3	0	0	0	x	0	x	0	x	0	13	x	x	mp	x	x	134	137	140	(u1*n2+u1*u2)*v	137	16	n1*u2*v	134	137	147	n1*n2*v	x	x	x	x
Fours	63	1	1	1	1	0	0	x	0	x	0	x	0	30	27	x	n1*mp	27	u1*mp	144	x	x	2*v	x	x	x	x	x	x	x	x	x	26	u1*gs
	64	1	1	1	0	1	0	x	0	x	0	x	0	34	28	x	n1*mp	34	u1*mp	144	x	x	2*v	x	x	x	x	x	x	x	x	x	26	u1*gs
	65	1	1	1	0	0	1	x	0	x	0	x	0	35	32	x	n1*mp	32	u1*mp	144	x	x	2*v	x	x	x	x	x	x	x	x	x	26	u1*gs
	66	1	1	0	2	0	0	119	(u1*n2*1/3)*1/3*h	117	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	111	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	36	x	x	mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	27	u1*gp	x	x
	67	1	1	0	0	2	0	125	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	123	(u1*n2*1/3)*1/3*h	112	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	40	x	x	n1*mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	28	u1*gp	x	x
	68	1	1	0	0	0	2	131	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	129	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	113	(u1*n2*1/3)*1/3*h	41	38	x	n1*mp	41	u1*mp	143	x	x	2*v	x	x	x	x	x	x	x	29	u1*gp	x	x
	69	0	1	1	2	0	0	111	(u1*n2*1/3)*1/3*h	115	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	114	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	42	39	x	n1*mp	42	u1*mp	143	x	x	2*v	x	x	x	x	x	x	x	33	u1*gp	x	x
	70	0	1	1	0	2	0	112	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	121	(u1*n2*1/3)*1/3*h	120	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	40	x	x	mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	34	u1*gp	x	x
	71	0	1	1	0	0	2	113	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	127	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	126	(u1*n2*1/3)*1/3*h	44	x	x	n1*mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	35	u1*gp	x	x
	72	1	0	1	2	0	0	118	(u1*n2*1/3)*1/3*h	111	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	116	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	36	x	x	n1*mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	30	u1*gp	x	x
	73	1	0	1	0	2	0	124	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	112	(u1*n2*1/3)*1/3*h	122	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	43	37	x	n1*mp	37	u1*mp	143	x	x	2*v	x	x	x	x	x	x	x	31	u1*gp	x	x
	74	1	0	1	0	0	2	130	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	113	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	128	(u1*n2*1/3)*1/3*h	44	x	x	mp	x	x	143	x	x	2*v	x	x	x	x	x	x	x	32	u1*gp	x	x
	75	2	1	0	1	0	0	x	0	x	0	x	0	51	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	50	u1*gs
	76	2	1	0	0	1	0	x	0	x	0	x	0	28	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	50	u1*gs
	77	2	1	0	0	0	1	x	0	x	0	x	0	29	53	x	n1*mp	53	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	x	x	50	u1*gs
	78	2	0	1	1	0	0	x	0	x	0	x	0	51	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	49	u1*gs
	79	2	0	1	0	1	0	x	0	x	0	x	0	31	52	x	n1*mp	31	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	x	x	49	u1*gs
	80	2	0	1	0	0	1	x	0	x	0	x	0	32	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	49	u1*gs
	81	0	2	1	1	0	0	x	0	x	0	x	0	33	54	x	n1*mp	54	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	x	x	46	u1*gs
	82	0	2	1	0	1	0	x	0	x	0	x	0	55	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	46	u1*gs
	83	0	2	1	0	0	1	x	0	x	0	x	0	35	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	46	u1*gs
	84	0	1	2	1	0	0	x	0	x	0	x	0	57	33	x	n1*mp	33	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	x	x	45	u1*gs
	85	0	1	2	0	1	0	x	0	x	0	x	0	34	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	45	u1*gs
	86	0	1	2	0	0	1	x	0	x	0	x	0	59	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	45	u1*gs
	87	1	2	0	1	0	0	x	0	x	0	x	0	27	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	48	u1*gs
	88	1	2	0	0	1	0	x	0	x	0	x	0	55	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	48	u1*gs
	89	1	2	0	0	0	1	x	0	x	0	x	0	56	29	x	n1*mp	29	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	x	x	48	u1*gs
	90	1	0	2	1	0	0	x	0	x	0	x	0	30	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	47	u1*gs
	91	1	0	2	0	1	0	x	0	x	0	x	0	58	31	x	n1*mp	58	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	x	x	47	u1*gs
	92	1	0	2	0	0	1	x	0	x	0	x	0	59	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	x	x	47	u1*gs
	93	2	0	0	2	0	0	132	(u1*n2*1/3)*1/3*h	119	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	118	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	x	x	x	x	x	x	145	x	x	2*v	x	x	x	x	x	x	x	51	u1*gp	x	x
	94	2	0	0	0	2	0	135	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	125	(u1*n2*1/3)*1/3*h	124	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	37	x	x	n1*mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	52	u1*gp	x	x
	95	2	0	0	0	0	2	138	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	131	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	130	(u1*n2*1/3)*1/3*h	38	x	x	mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	53	u1*gp	x	x
	96	0	2	0	2	0	0	117	(u1*n2*1/3)*1/3*h	133	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	115	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	39	x	x	mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	54	u1*gp	x	x
	97	0	2	0	0	2	0	123	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	136	(u1*n2*1/3)*1/3*h	121	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	x	x	x	x	x	x	145	x	x	2*v	x	x	x	x	x	x	x	55	u1*gp	x	x
	98	0	2	0	0	0	2	129	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	139	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	127	(u1*n2*1/3)*1/3*h	41	x	x	n1*mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	56	u1*gp	x	x
	99	0	0	2	2	0	0	116	(u1*n2*1/3)*1/3*h	114	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	134	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	42	x	x	n1*mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	57	u1*gp	x	x
	100	0	0	2	0	2	0	122	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	120	(u1*n2*1/3)*1/3*h	137	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	43	x	x	mp	x	x	145	x	x	2*v	x	x	x	x	x	x	x	58	u1*gp	x	x
	101	0	0	2	0	0	2	128	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*2/3)*1/3*h	126	(u1*n2*1/3 + n1*u2*1/2 + u1*u2*1/3)*1/3*h	140	(u1*n2*1/3)*1/3*h	x	x	x	x	x	x	145	x	x	2*v	x	x	x	x	x	x	x	59	u1*gp	x	x
	102	3	0	0	1	0	0	x	0	x	0	x	0	x	x	x	x	x	x	147	x	x	2*v	x	x	x	x	x	x	x	x	x	60	u1*gs
	103	3	0	0	0	1	0	x	0	x	0	x	0	52	x	x	mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	x	x	60	u1*gs
	104	3	0	0	0	0	1	x	0	x	0	x	0	53	x	x	n1*mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	x	x	60	u1*gs
	105	0	3	0	1	0	0	x	0	x	0	x	0	54	x	x	n1*mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	x	x	61	u1*gs
	106	0	3	0	0	1	0	x	0	x	0	x	0	x	x	x	x	x	x	147	x	x	2*v	x	x	x	x	x	x	x	x	x	61	u1*gs
	107	0	3	0	0	0	1	x	0	x	0	x	0	56	x	x	mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	x	x	61	u1*gs
	108	0	0	3	1	0	0	x	0	x	0	x	0	57	x	x	mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	x	x	62	u1*gs
	109	0	0	3	0	1	0	x	0	x	0	x	0	58	x	x	n1*mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	x	x	62	u1*gs
	110	0	0	3	0	0	1	x	0	x	0	x	0	x	x	x	x	x	x	147	x	x	2*v	x	x	x	x	x	x	x	x	x	62	u1*gs
Fives	111	1	1	1	2	0	0	x	0	x	0	x	0	72	66	x	n1*mp	72	u1*mp	144	x	x	2*v	x	x	x	x	x	x	x	63	u1*gp	x	x
	112	1	1	1	0	2	0	x	0	x	0	x	0	70	67	x	n1*mp	67	u1*mp	144	x	x	2*v	x	x	x	x	x	x	x	64	u1*gp	x	x
	113	1	1	1	0	0	2	x	0	x	0	x	0	71	74	x	n1*mp	71	u1*mp	144	x	x	2*v	x	x	x	x	x	x	x	65	u1*gp	x	x
	114	0	1	2	2	0	0	x	0	x	0	x	0	99	69	x	n1*mp	99	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	84	u1*gp	x	x
	115	0	2	1	2	0	0	x	0	x	0	x	0	69	96	x	n1*mp	69	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	81	u1*gp	x	x
	116	1	0	2	2	0	0	x	0	x	0	x	0	72	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	90	u1*gp	x	x
	117	1	2	0	2	0	0	x	0	x	0	x	0	66	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	87	u1*gp	x	x
	118	2	0	1	2	0	0	x	0	x	0	x	0	93	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	78	u1*gp	x	x
	119	2	1	0	2	0	0	x	0	x	0	x	0	93	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	75	u1*gp	x	x
	120	0	1	2	0	2	0	x	0	x	0	x	0	70	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	85	u1*gp	x	x
	121	0	2	1	0	2	0	x	0	x	0	x	0	97	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	82	u1*gp	x	x
	122	1	0	2	0	2	0	x	0	x	0	x	0	100	73	x	n1*mp	73	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	91	u1*gp	x	x
	123	1	2	0	0	2	0	x	0	x	0	x	0	97	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	88	u1*gp	x	x
	124	2	0	1	0	2	0	x	0	x	0	x	0	73	94	x	n1*mp	94	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	79	u1*gp	x	x
	125	2	1	0	0	2	0	x	0	x	0	x	0	67	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	76	u1*gp	x	x
	126	0	1	2	0	0	2	x	0	x	0	x	0	101	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	86	u1*gp	x	x
	127	0	2	1	0	0	2	x	0	x	0	x	0	71	x	x	n1*mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	83	u1*gp	x	x
	128	1	0	2	0	0	2	x	0	x	0	x	0	101	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	92	u1*gp	x	x
	129	1	2	0	0	0	2	x	0	x	0	x	0	98	68	x	n1*mp	98	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	89	u1*gp	x	x
	130	2	0	1	0	0	2	x	0	x	0	x	0	74	x	x	mp	x	x	146	x	x	2*v	x	x	x	x	x	x	x	80	u1*gp	x	x
	131	2	1	0	0	0	2	x	0	x	0	x	0	68	95	x	n1*mp	68	u1*mp	146	x	x	2*v	x	x	x	x	x	x	x	77	u1*gp	x	x
	132	3	0	0	2	0	0	x	0	x	0	x	0	x	x	x	x	x	x	147	x	x	2*v	x	x	x	x	x	x	x	102	u1*gp	x	x
	133	0	3	0	2	0	0	x	0	x	0	x	0	96	x	x	mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	105	u1*gp	x	x
	134	0	0	3	2	0	0	x	0	x	0	x	0	99	x	x	n1*mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	108	u1*gp	x	x
	135	3	0	0	0	2	0	x	0	x	0	x	0	94	x	x	n1*mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	103	u1*gp	x	x
	136	0	3	0	0	2	0	x	0	x	0	x	0	x	x	x	x	x	x	147	x	x	2*v	x	x	x	x	x	x	x	106	u1*gp	x	x
	137	0	0	3	0	2	0	x	0	x	0	x	0	100	x	x	mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	109	u1*gp	x	x
	138	3	0	0	0	0	2	x	0	x	0	x	0	95	x	x	mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	104	u1*gp	x	x
	139	0	3	0	0	0	2	x	0	x	0	x	0	98	x	x	n1*mp	x	x	147	x	x	2*v	x	x	x	x	x	x	x	107	u1*gp	x	x
	140	0	0	3	0	0	2	x	0	x	0	x	0	x	x	x	x	x	x	147	x	x	2*v	x	x	x	x	x	x	x	110	u1*gp	x	x
System Failures	141	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	u3*w	x	x	x	x	x	x	x	x	x	x	x
	142	1	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	2	x	x	u3*w	x	x	x	x	x	x	x	x	x	x	x
	143	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	8	x	x	u3*w	x	x	x	x	x	x	x	x	x	x	x
	144	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	26	x	x	u3*w	x	x	x	x	x	x	x	x	x	x	x
	145	2	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	11	x	x	u3*w	x	x	x	x	x	x	x	x	x	x	x
	146	2	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	50	x	x	u3*w	x	x	x	x	x	x	x	x	x	x	x
	147	3	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	60	x	x	u3*w	x	x	x	x	x	x	x	x	x	x	x